

**EduTech Innovators SPA**

**Jonathan Larraguibel, Daniel Morales y Danae Miranda**

**Eduardo Baeza**

**DESARROLLO FULLSTACK I\_011D**

# **Índice**

[**Índice 2**](#_30j0zll)

[**Diagrama de arquitectura de microservicios 3**](#_3znysh7)

[**Plan de pruebas 4**](#_2et92p0)

[Proceso de Versionado y Subida del Proyecto a GitHub 38](#_9anl4bporar0)

[Clonación del Repositorio 38](#_6mvpm26y5nqv)

[Exploración de Ramas Remotas 38](#_mk5n9c5l9wxt)

[Cambio y Configuración de Rama de Trabajo 38](#_bb92zo4siqr1)

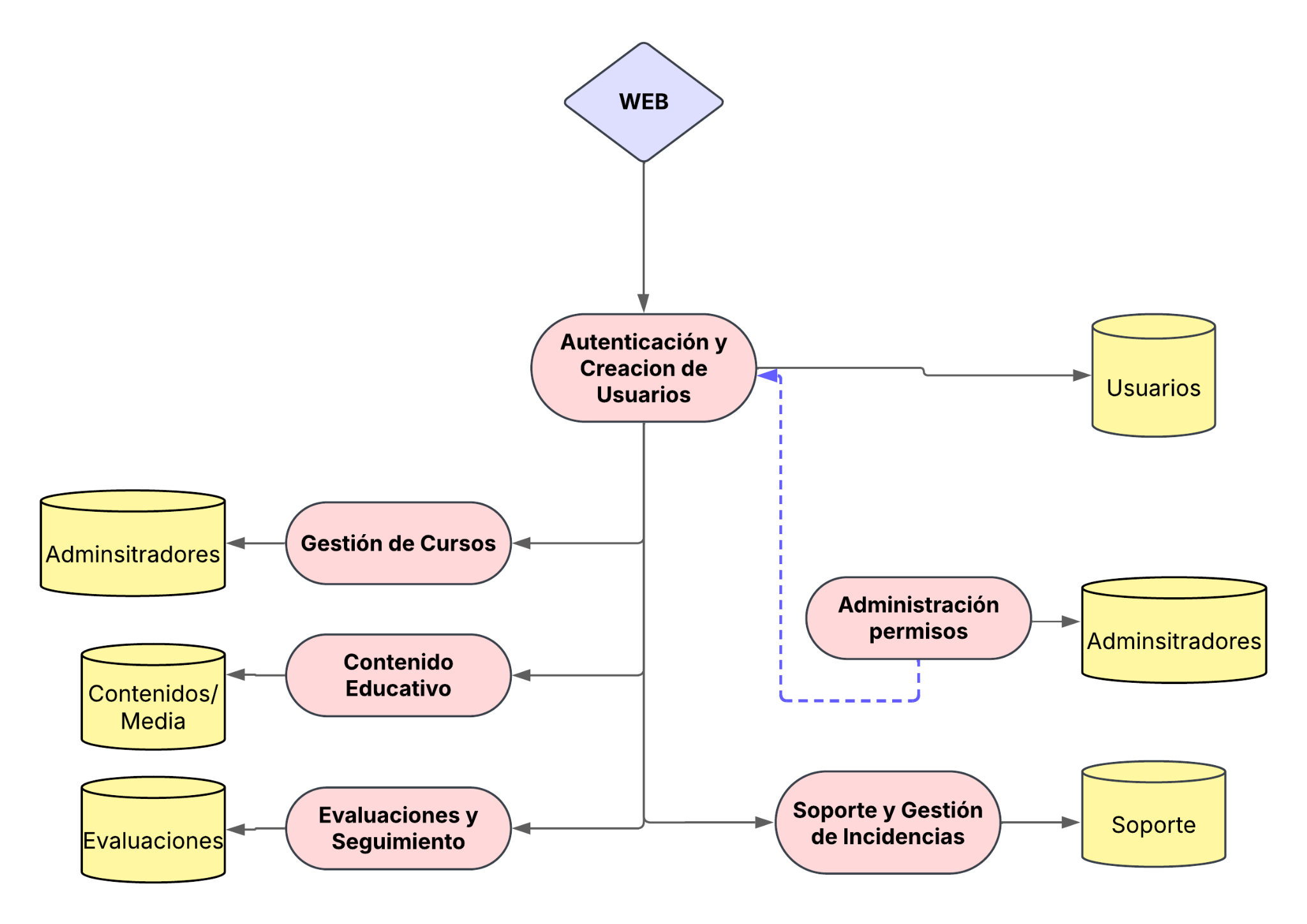
[Revisión del Estado de Archivos 39](#_kgqctb9lepmq)

[Confirmación de Cambios (Commit) 39](#_tq2i7zpjlgnt)

[Subida de Cambios al Repositorio Remoto (Push) 39](#_zgn755uhs09g)

[**Conclusión 44**](#)

# **Diagrama de arquitectura de microservicios**



# **Plan de pruebas**

Como equipo de desarrollo del proyecto EduTech Innovators, consideramos fundamental incorporar herramientas especializadas para garantizar la calidad y fiabilidad del sistema que construimos. En este sentido, decidimos implementar un conjunto de frameworks ampliamente utilizados en el desarrollo backend con Spring Boot, los cuales nos permitieron aplicar tanto pruebas unitarias como pruebas de integración en los distintos componentes de nuestros microservicios.

**Herramientas seleccionadas y fundamentos de elección**

1. ***JUnit 5 (Jupiter)***

Seleccionamos JUnit 5 como nuestro framework principal para la creación de pruebas unitarias debido a su integración directa con Spring Boot y su sintaxis moderna basada en anotaciones. Nos permitió escribir pruebas organizadas y claras, enfocadas en verificar el comportamiento de los métodos de nuestros servicios, como EvaluacionServiceImpl o PreguntaServiceImpl, de forma independiente del resto del sistema.

2. ***Mockito***

Implementamos Mockito para simular el comportamiento de los repositorios, lo cual nos permitió enfocarnos exclusivamente en la lógica de negocio durante las pruebas. Mediante el uso de anotaciones como @Mock y @InjectMocks, pudimos evitar el acceso real a la base de datos y generar pruebas controladas, eficientes y repetibles. Esta herramienta fue clave para asegurar que nuestras pruebas fueran verdaderamente unitarias.

3. ***Spring Boot Test***

Optamos por utilizar el módulo spring-boot-starter-test como base para estructurar nuestro entorno de pruebas. Gracias a este módulo y al uso de anotaciones como @SpringBootTest, pudimos levantar un contexto de aplicación completo, simulando de manera realista el comportamiento de nuestros controladores y servicios, lo que resultó esencial en las pruebas de integración.

4. ***MockMvc***

Para la verificación de los controladores REST, empleamos MockMvc, que nos permitió simular peticiones HTTP a nuestros endpoints sin necesidad de desplegar el servidor. De esta manera, pudimos validar que las rutas definidas como /api/evaluaciones o /api/respuestas-estudiante respondieran correctamente, devolviendo los datos esperados y los códigos HTTP adecuados (200 OK, 404 Not Found, 201 Created, etc.).

5. ***ObjectMapper (Jackson)***

Finalmente, utilizamos la clase ObjectMapper para la conversión de objetos Java a JSON y viceversa. Esta herramienta fue fundamental al momento de simular solicitudes POST y PUT, ya que nos permitió enviar cuerpos de datos realistas en nuestras pruebas y validar que los controladores procesarán correctamente la información recibida.

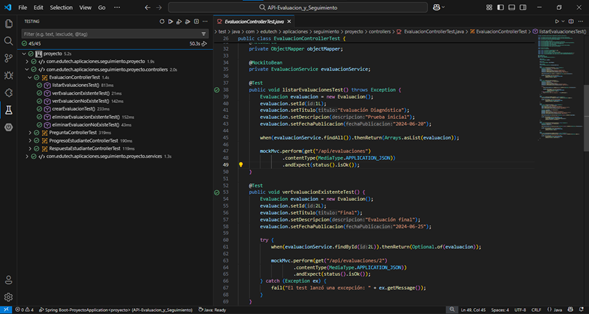
**Impacto en la calidad del proyecto**

Gracias a la implementación de estas herramientas, logramos construir un sistema confiable y bien estructurado. Las pruebas nos permitieron identificar y corregir errores en etapas tempranas del desarrollo, lo que facilitó el mantenimiento del código y mejoró significativamente la estabilidad de los servicios.

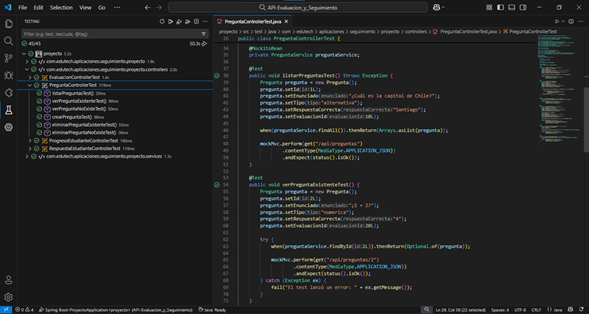
Además, este enfoque nos permitió mantener un alto grado de modularidad y desacoplamiento, ya que cada componente fue probado de manera aislada antes de ser integrado. El trabajo colaborativo con estas herramientas también reforzó nuestras habilidades como equipo en el uso de buenas prácticas para proyectos basados en microservicios.

**API-Evaluaciones\_y\_Seguimiento**

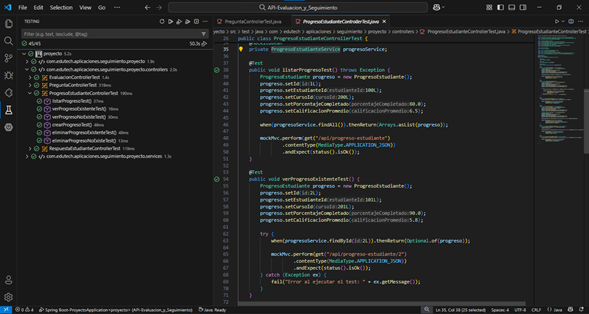
EvaluacionControllerTest:



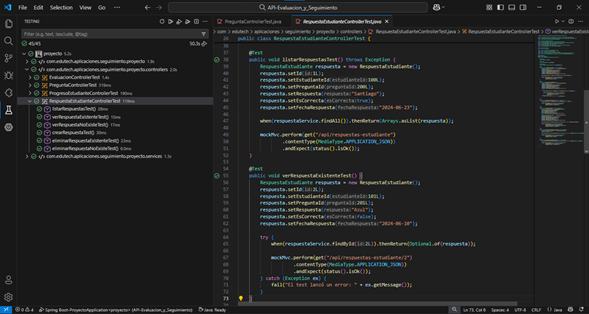
PreguntaControllerTest:



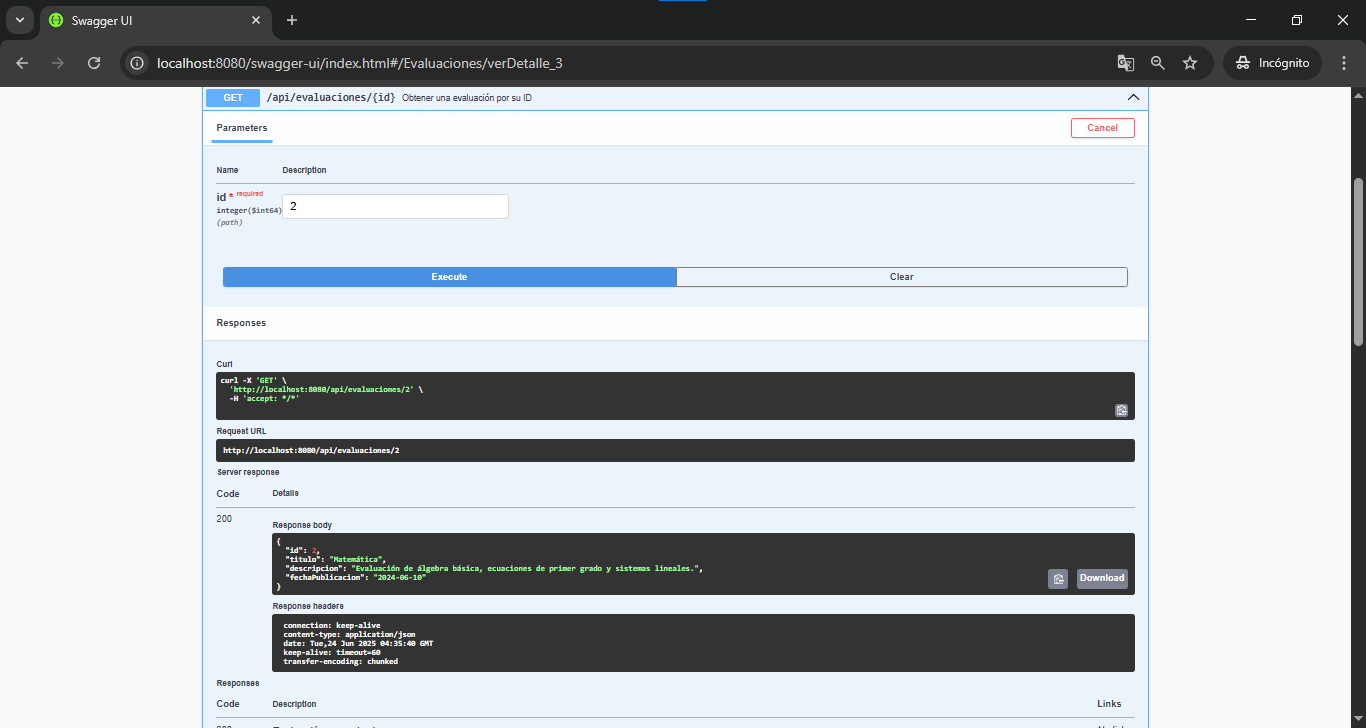
ProgresoEstudianteControllerTest:



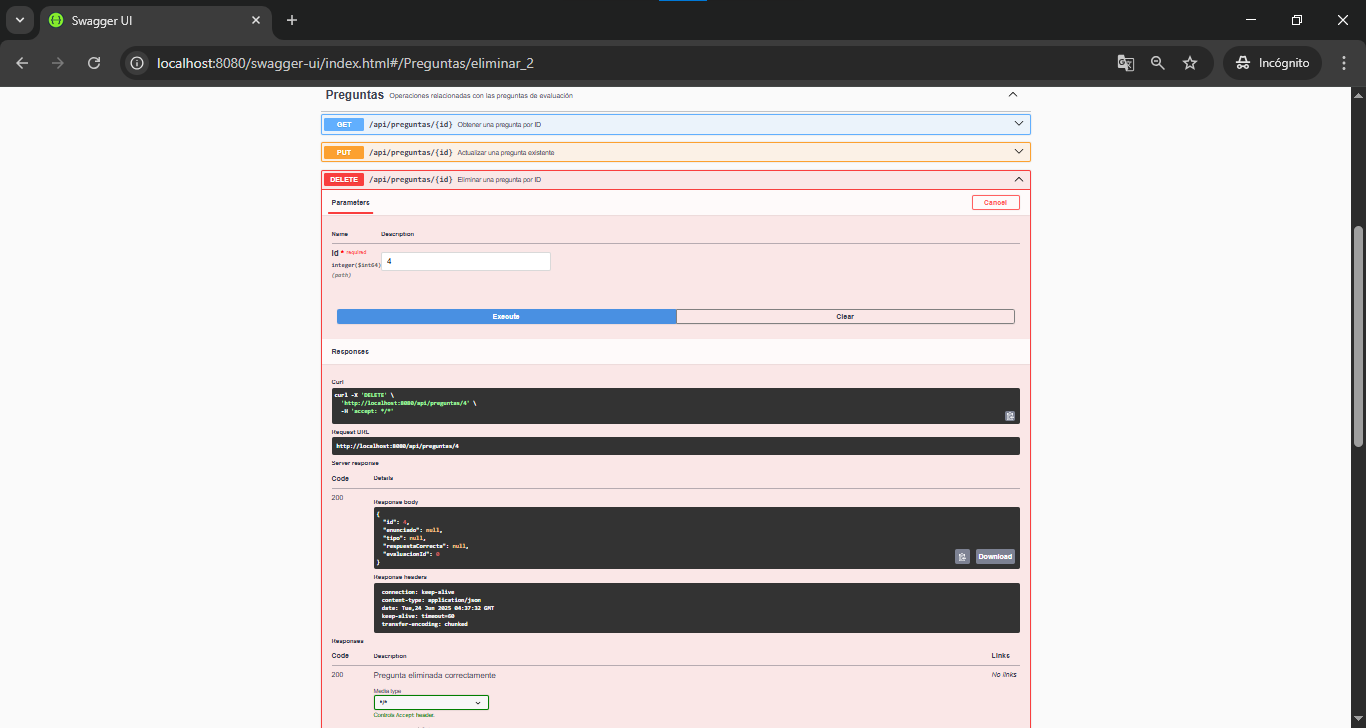
RespuestaEstudianteControllerTest:



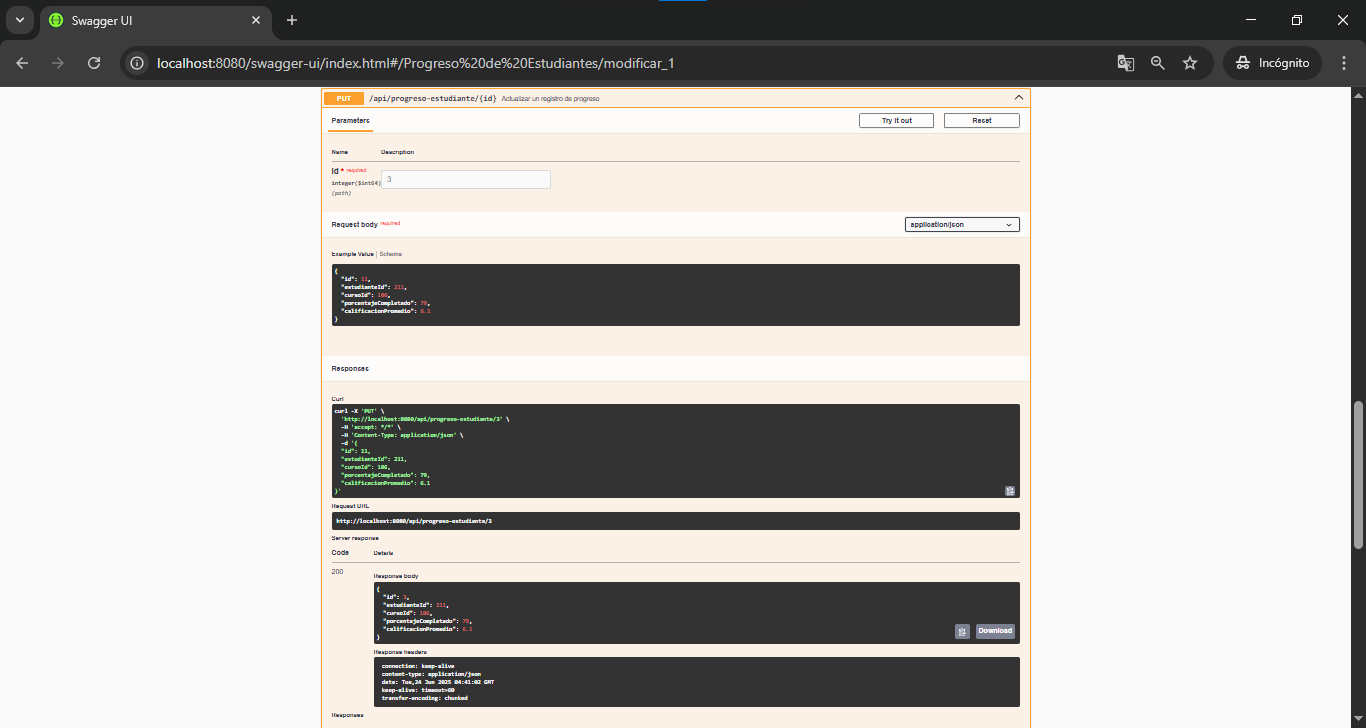
EvaluacionControllerTest:

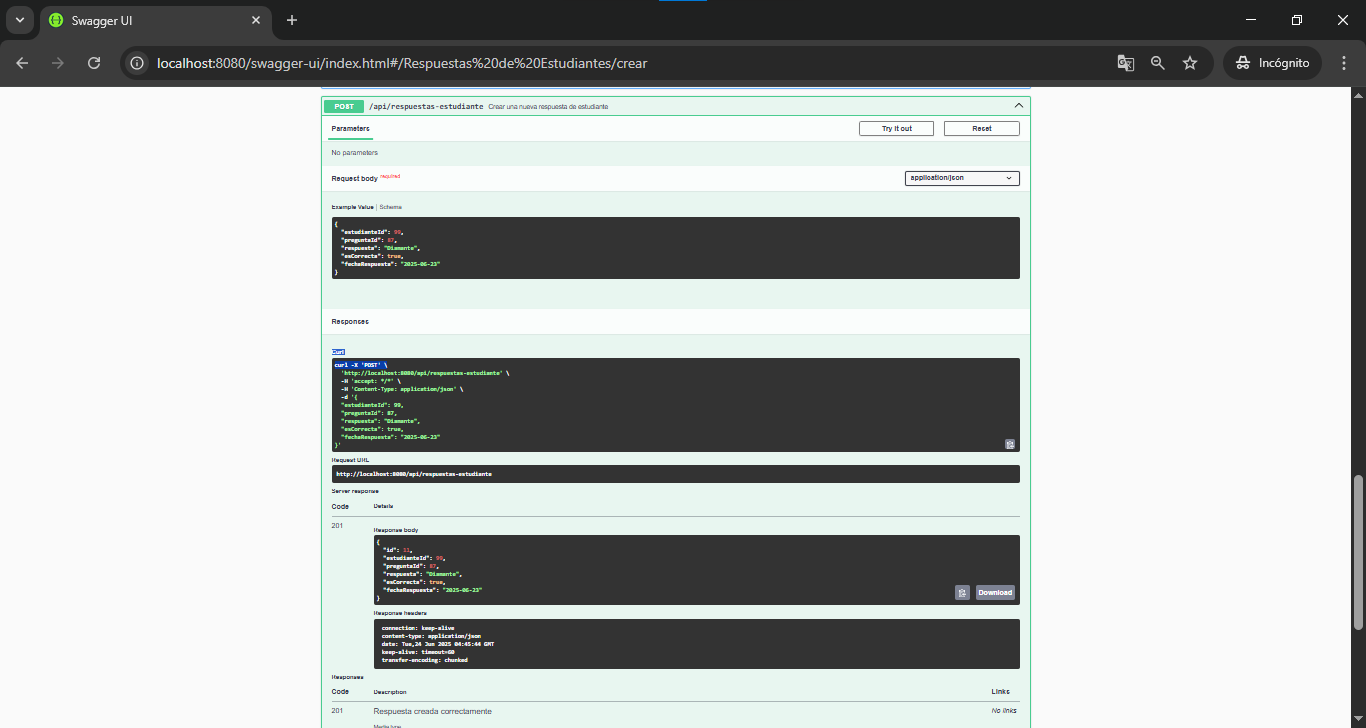


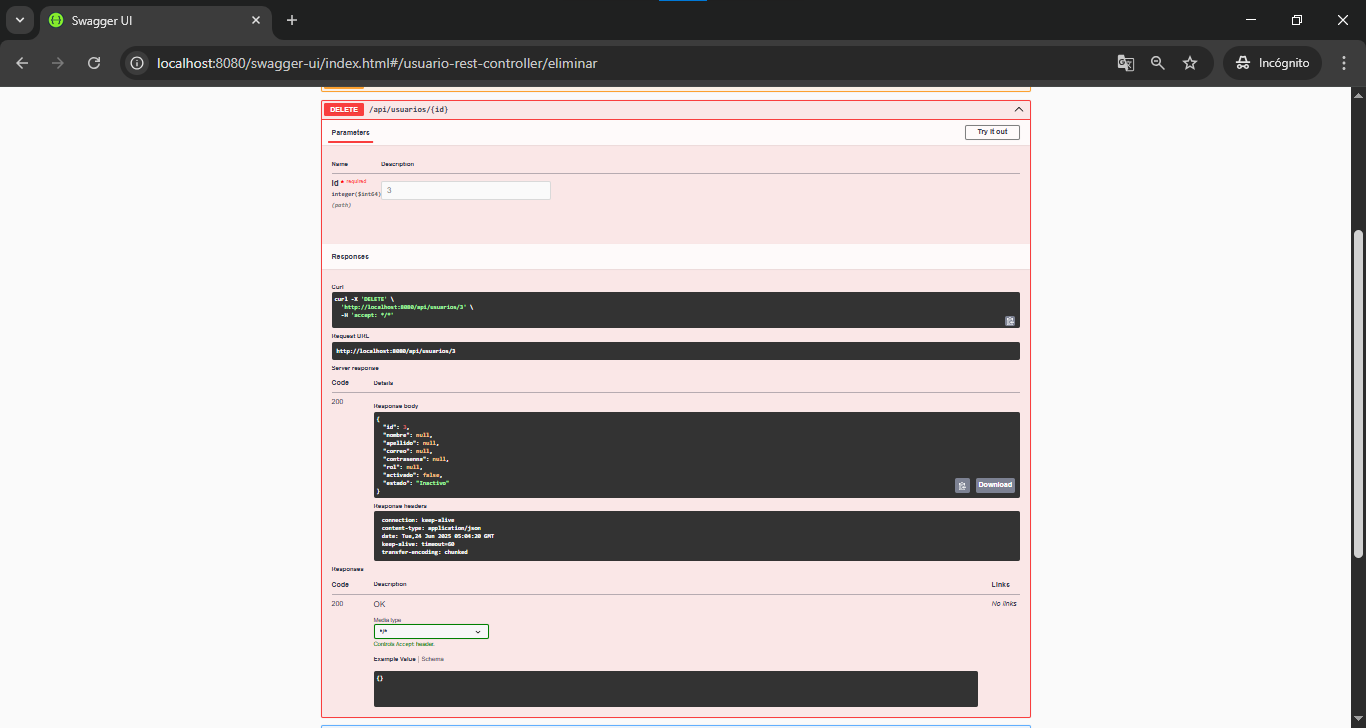
PreguntaControllerTest:

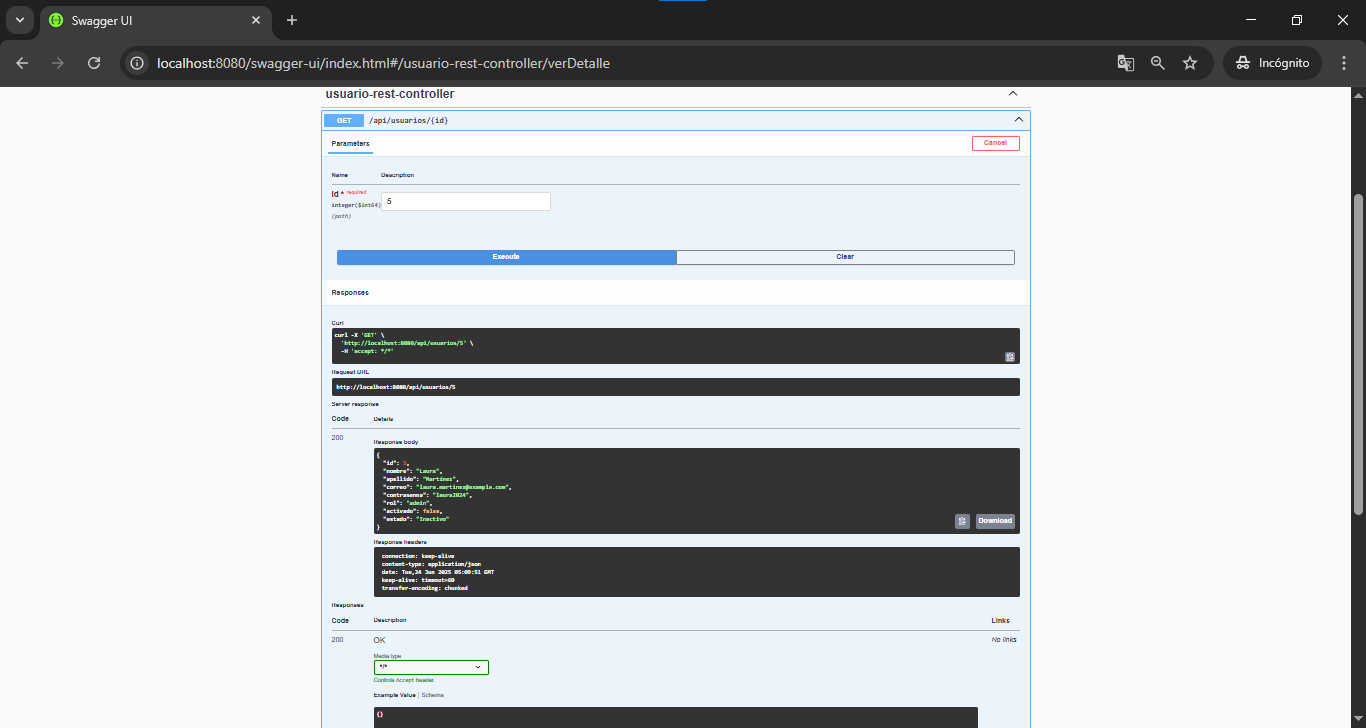


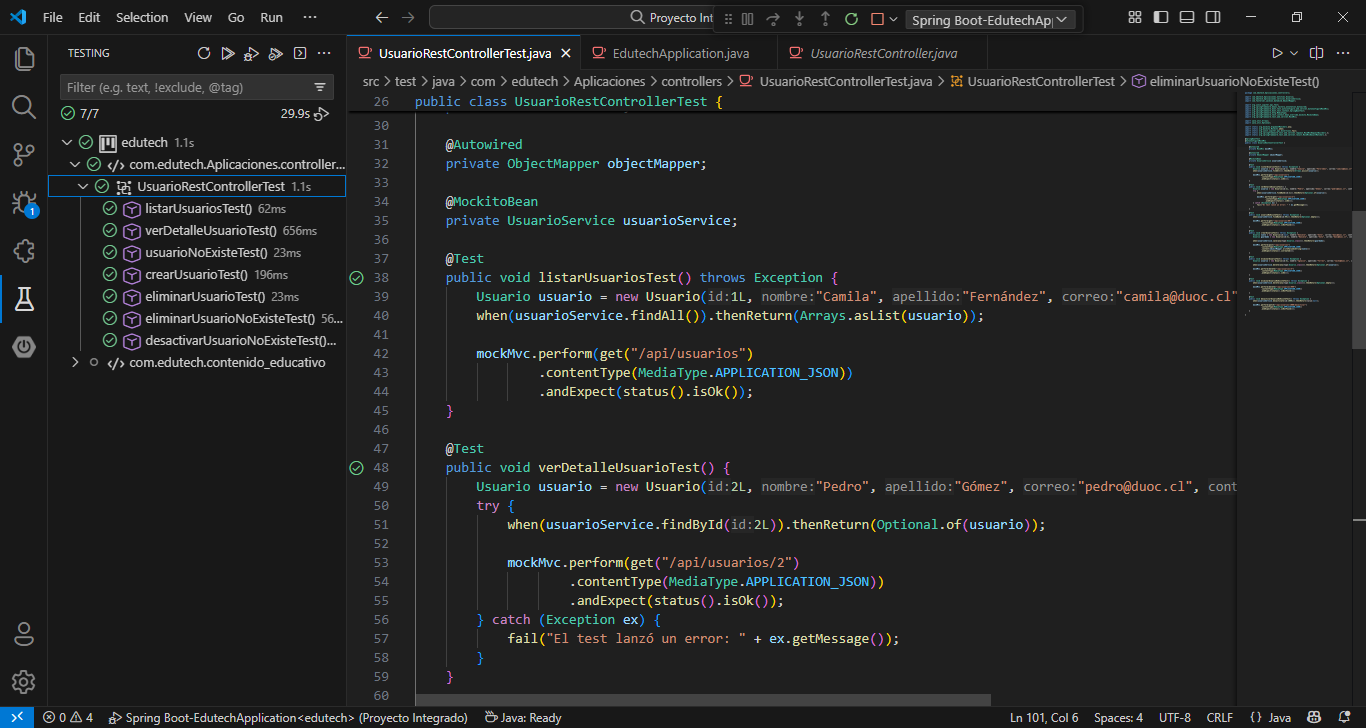
ProgresoEstudianteControllerTest:



RespuestaEstudianteControllerTest:  


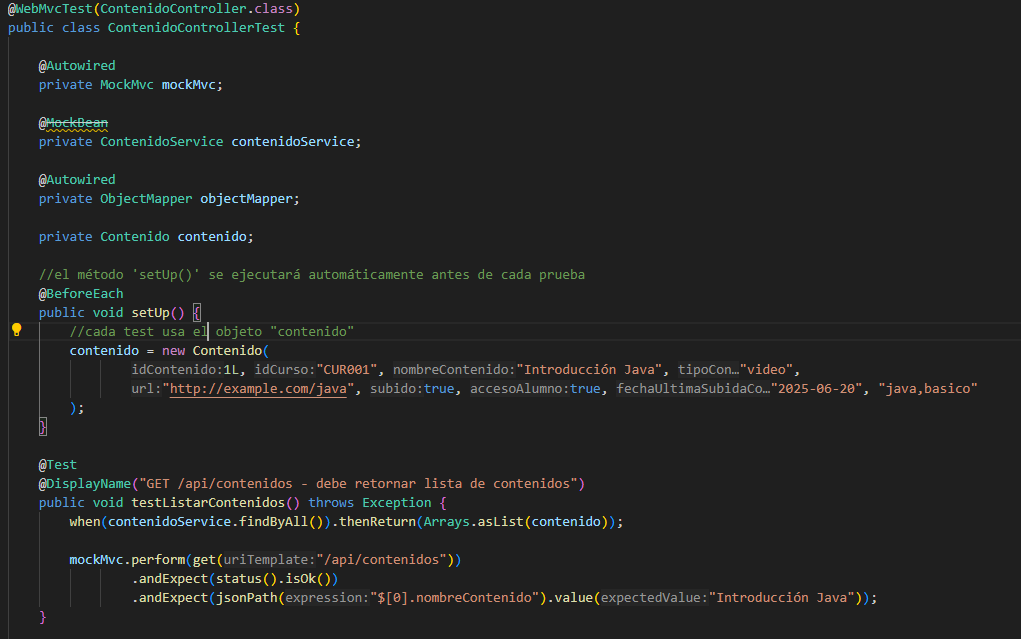
UsuarioResControllerTest:  


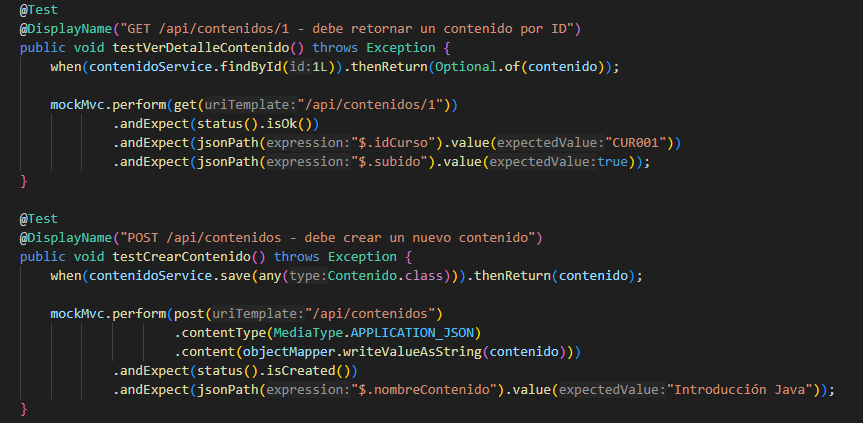




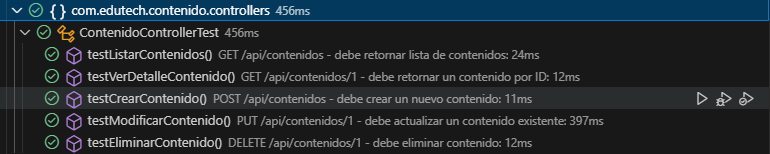
**API-Contenido Educativo**

Pruebas de integración:

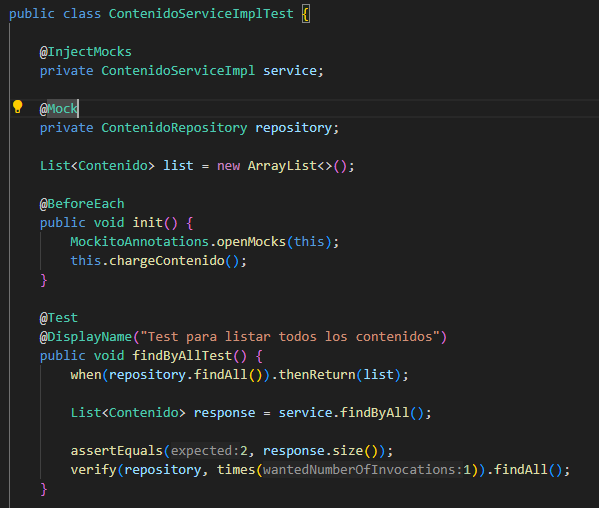


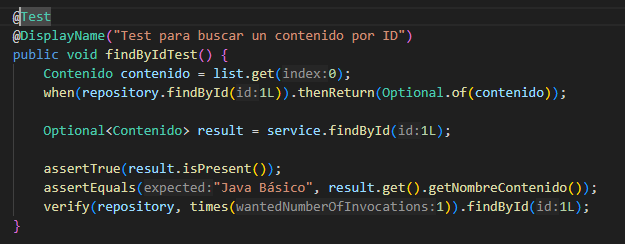


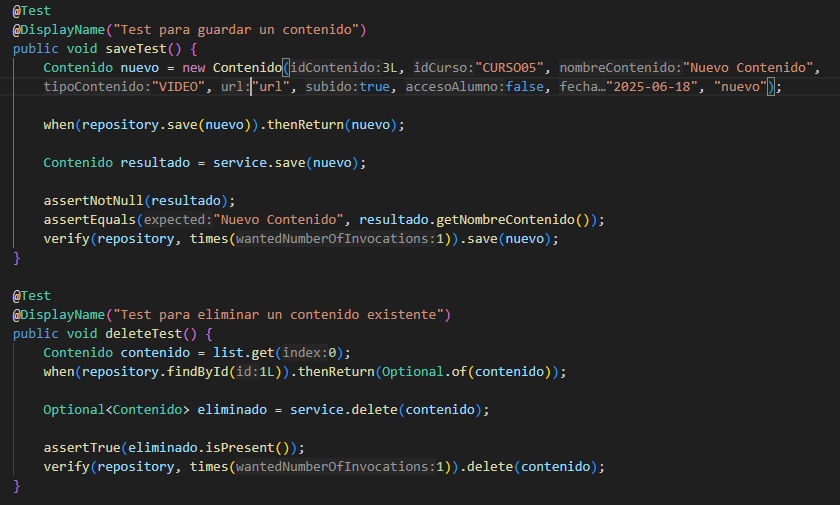




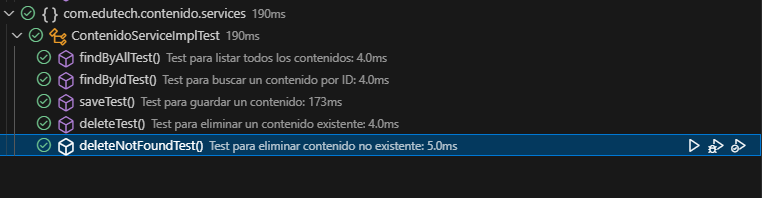
Pruebas unitarias:

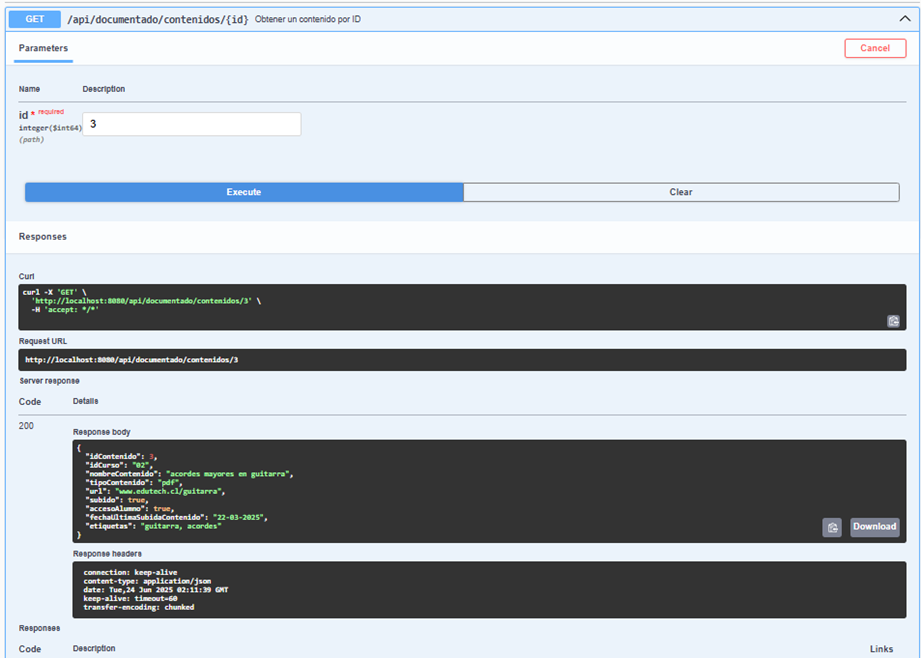


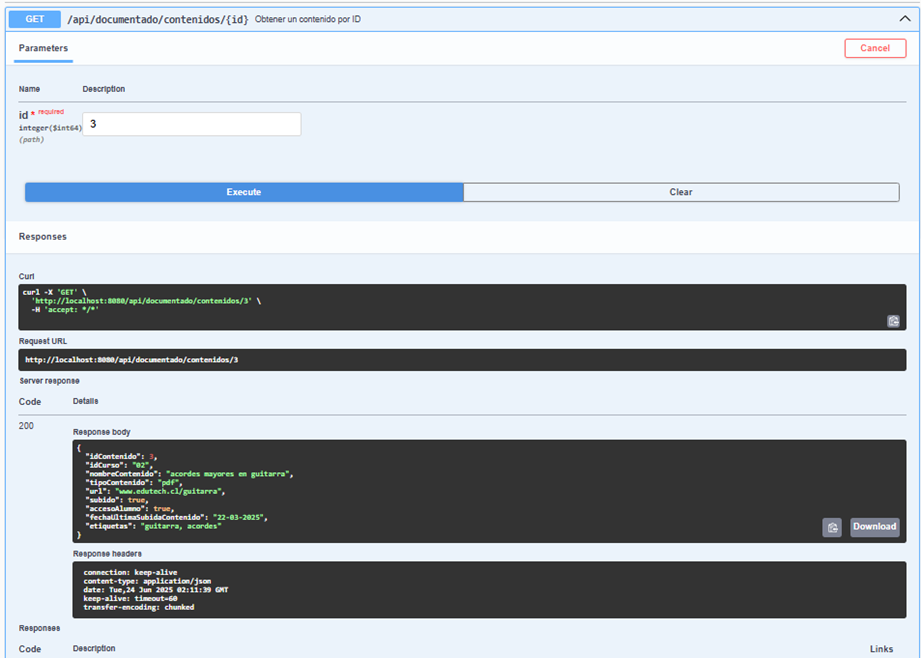


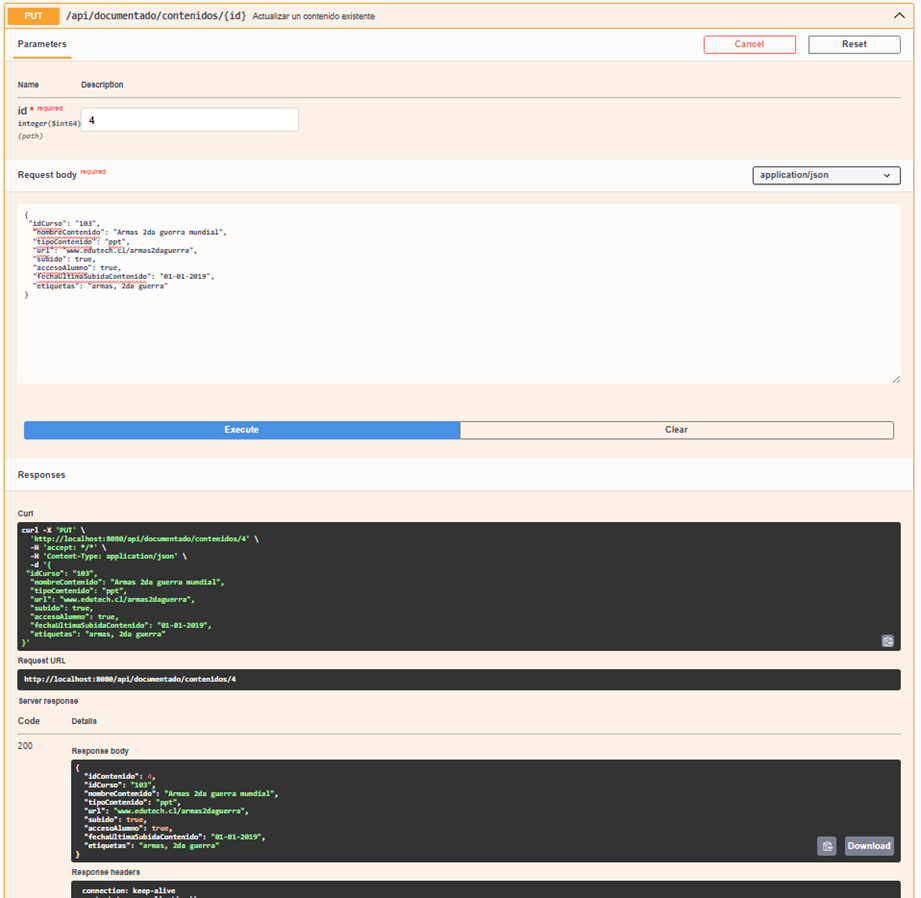


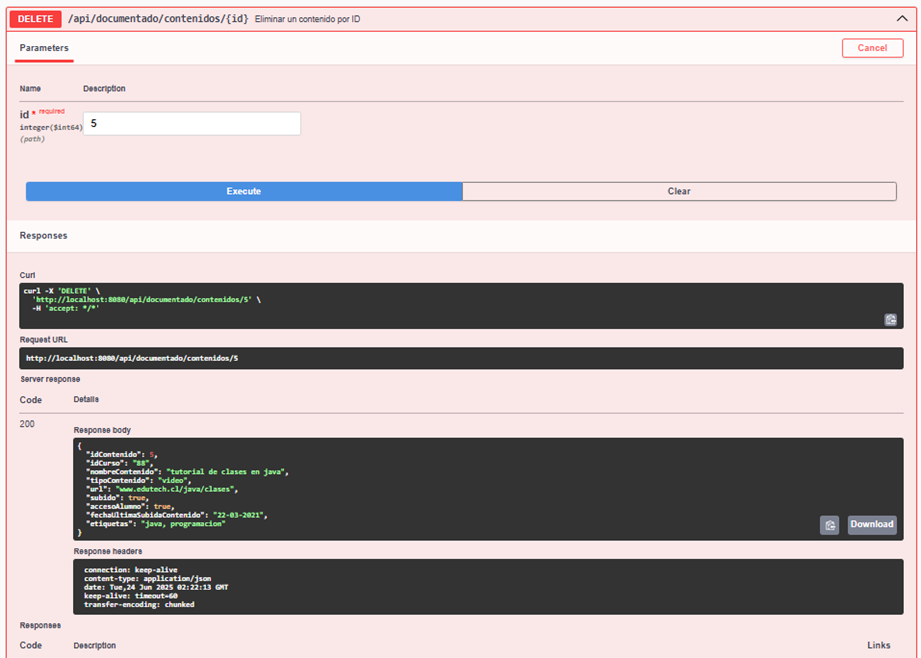


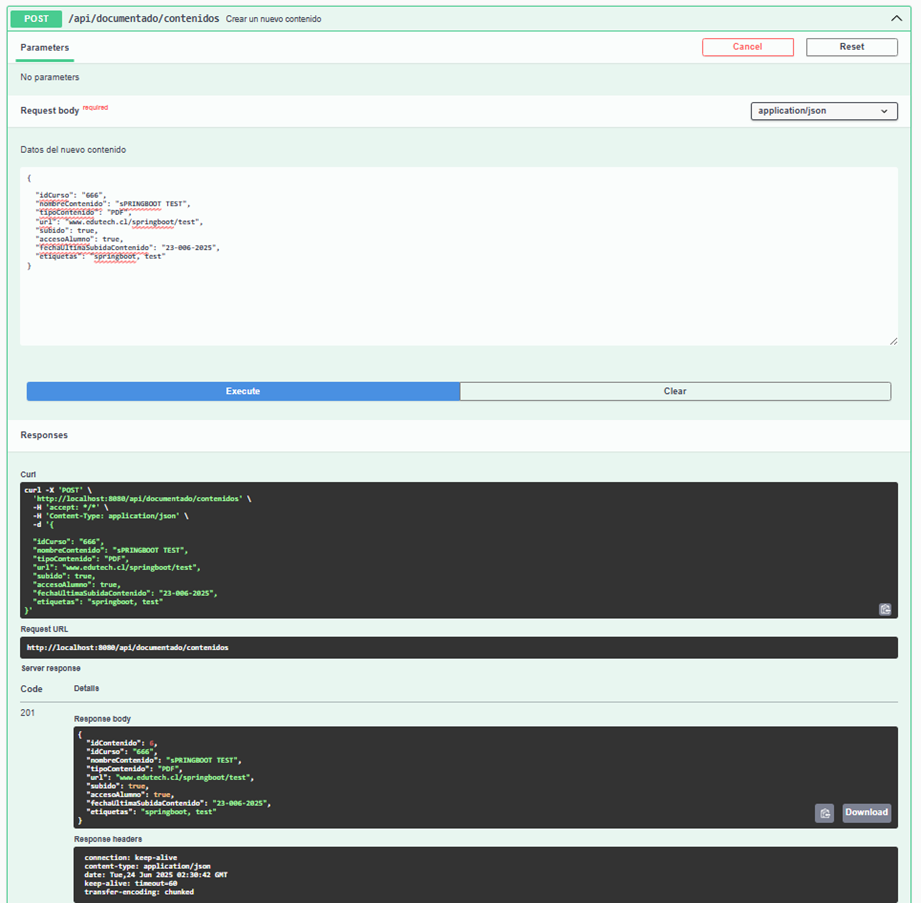








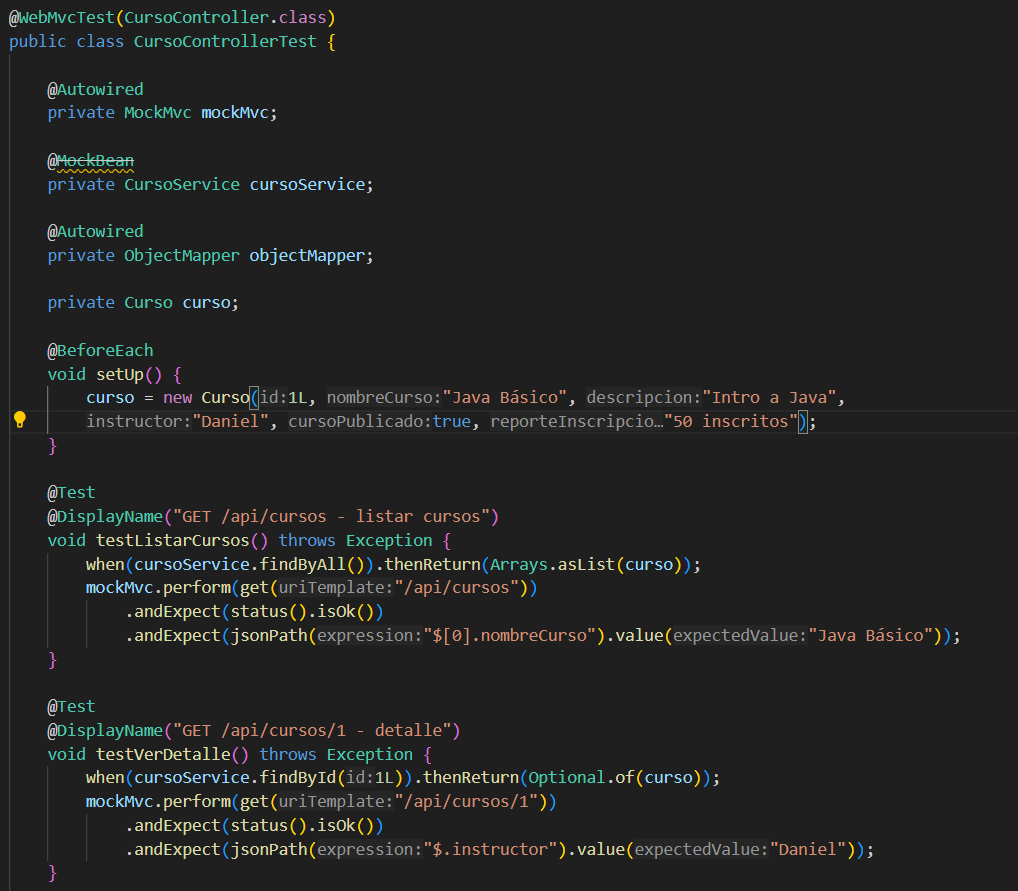






**API-Curso**

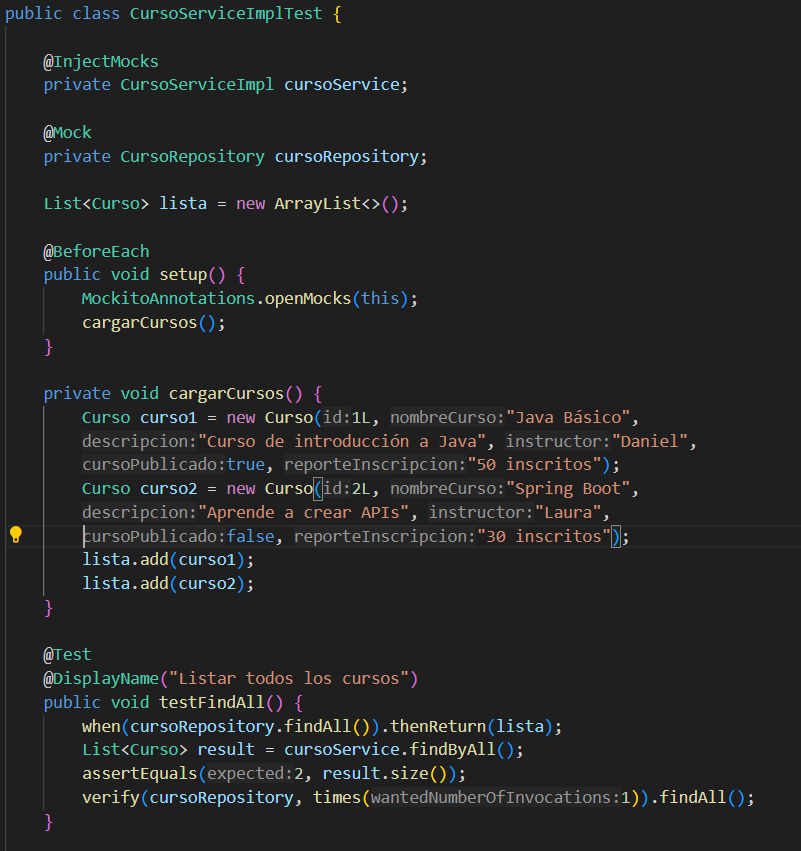
prueba Integración:

****

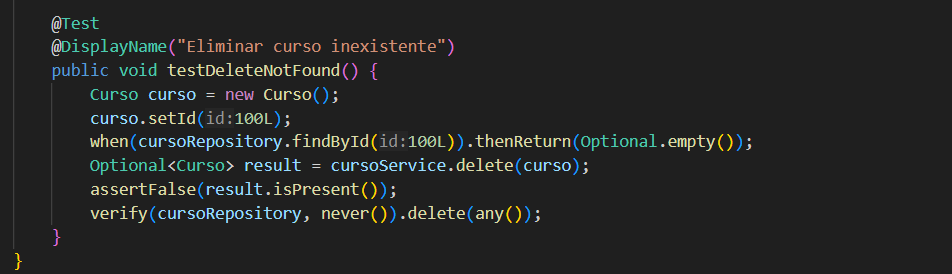
****

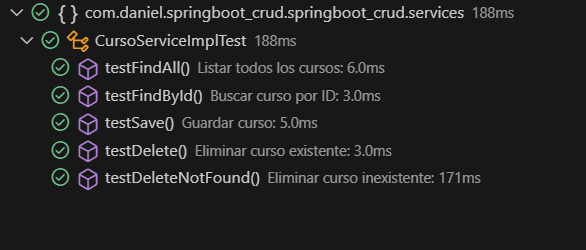
****

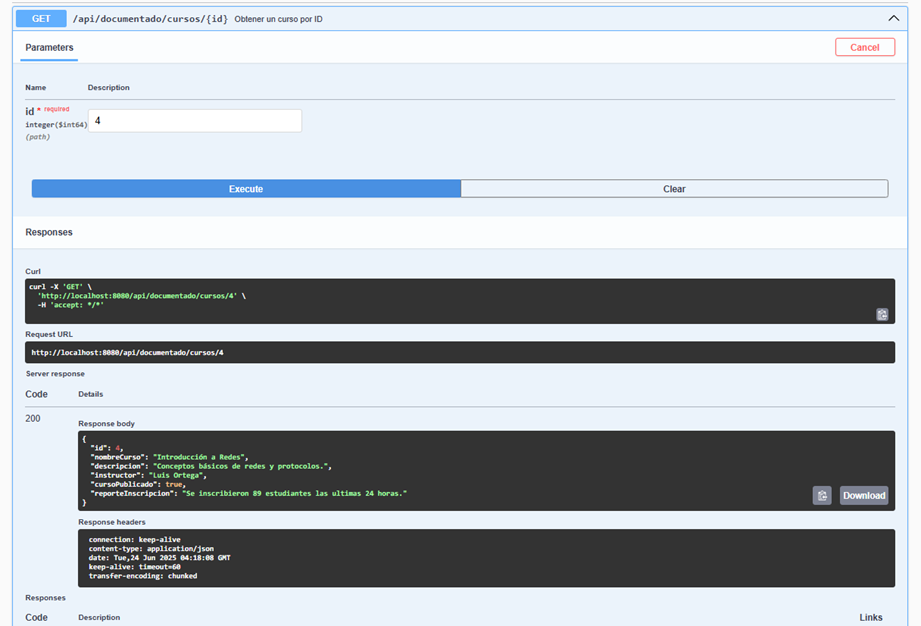
pruebas unitarias:

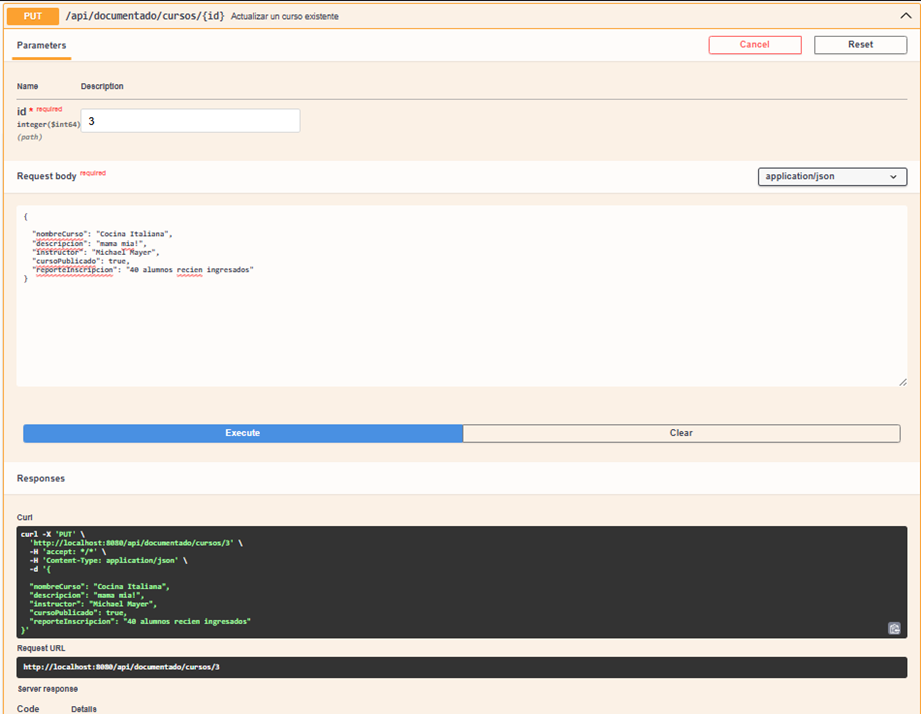
****

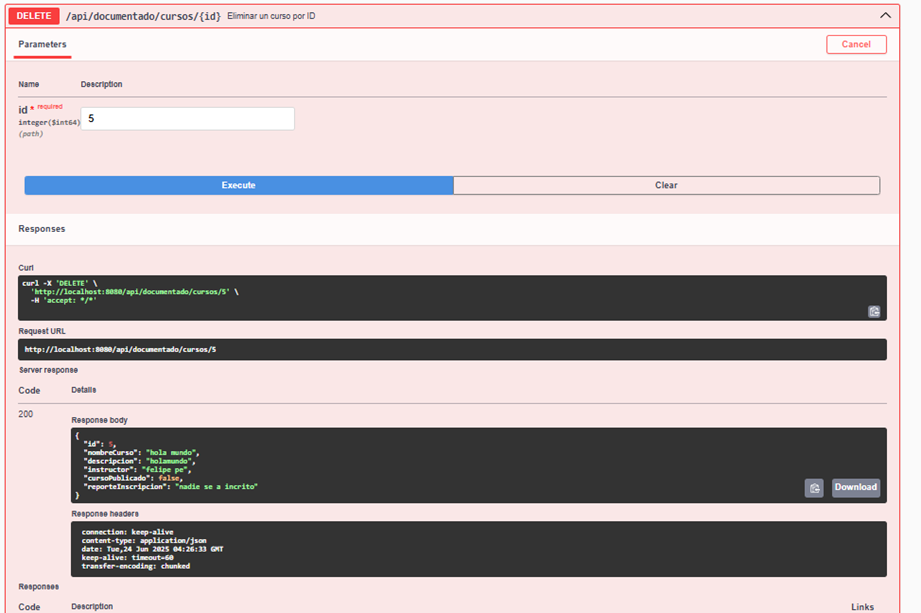
****

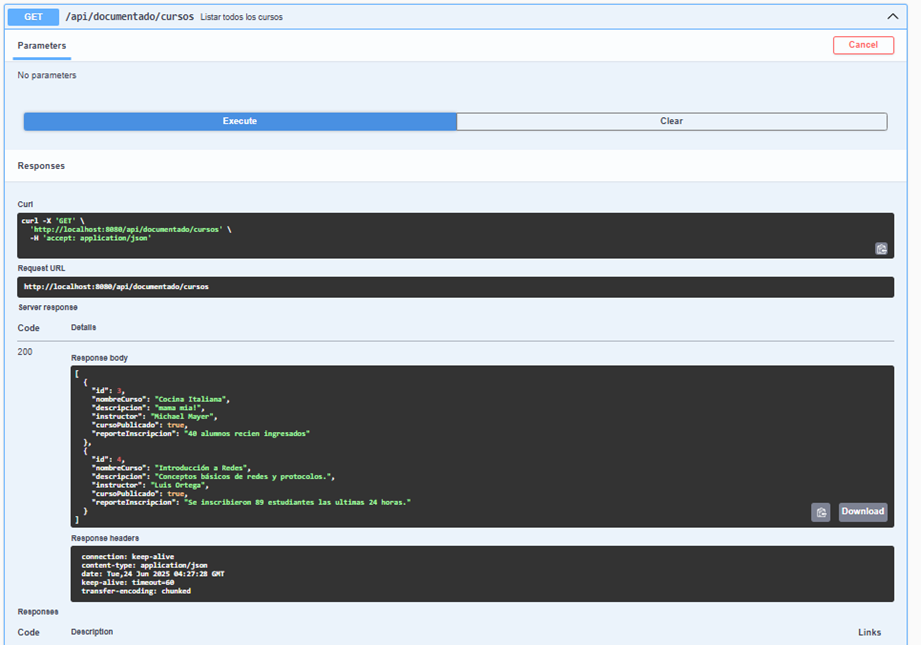
****

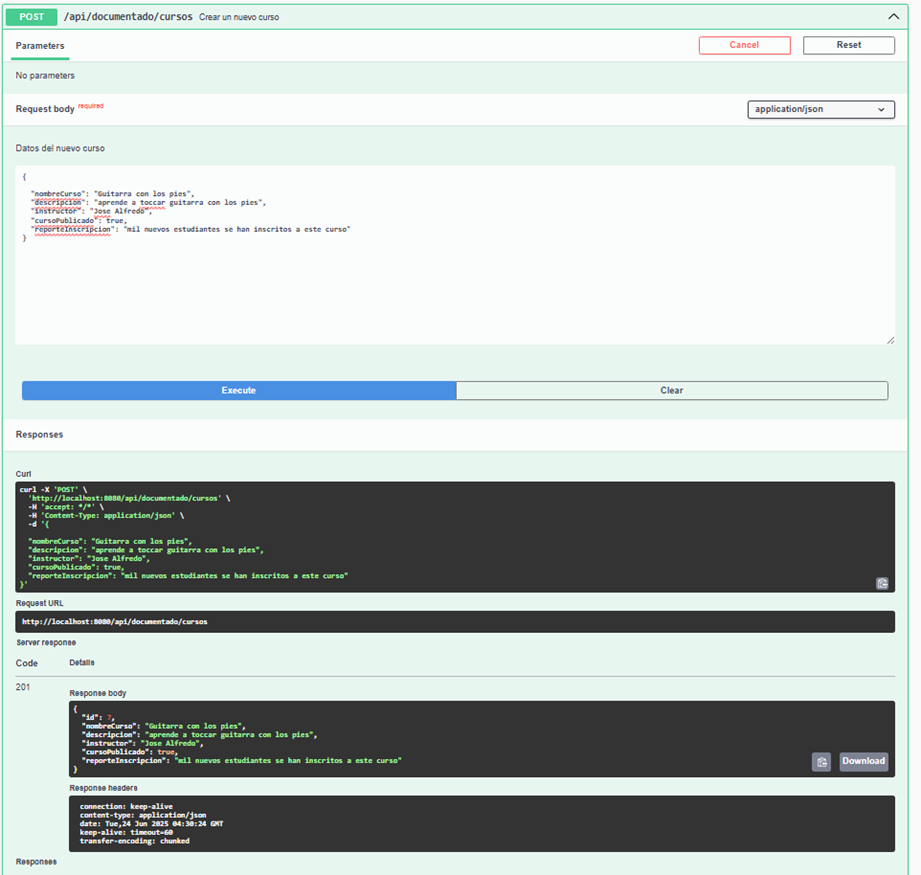
****

****

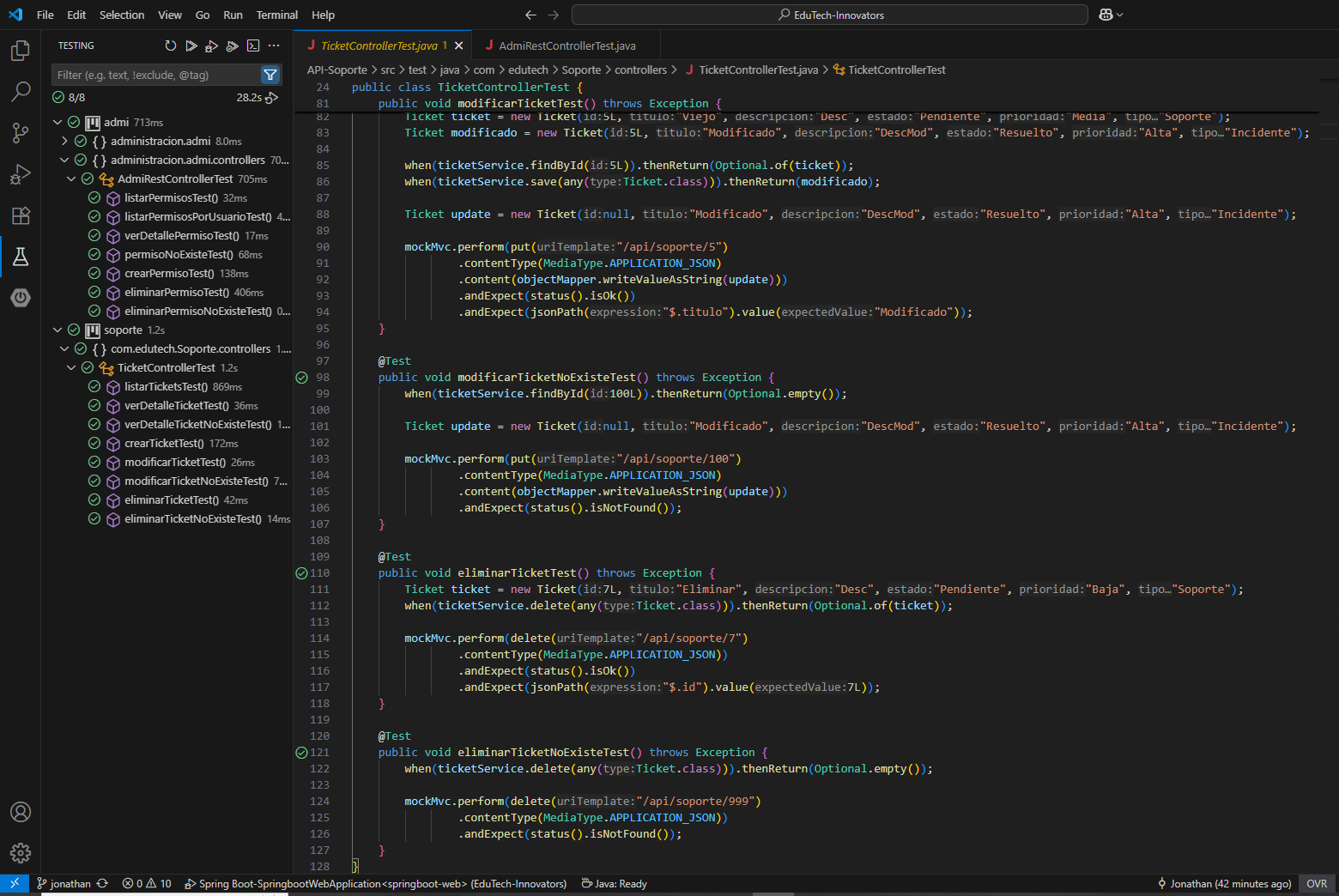
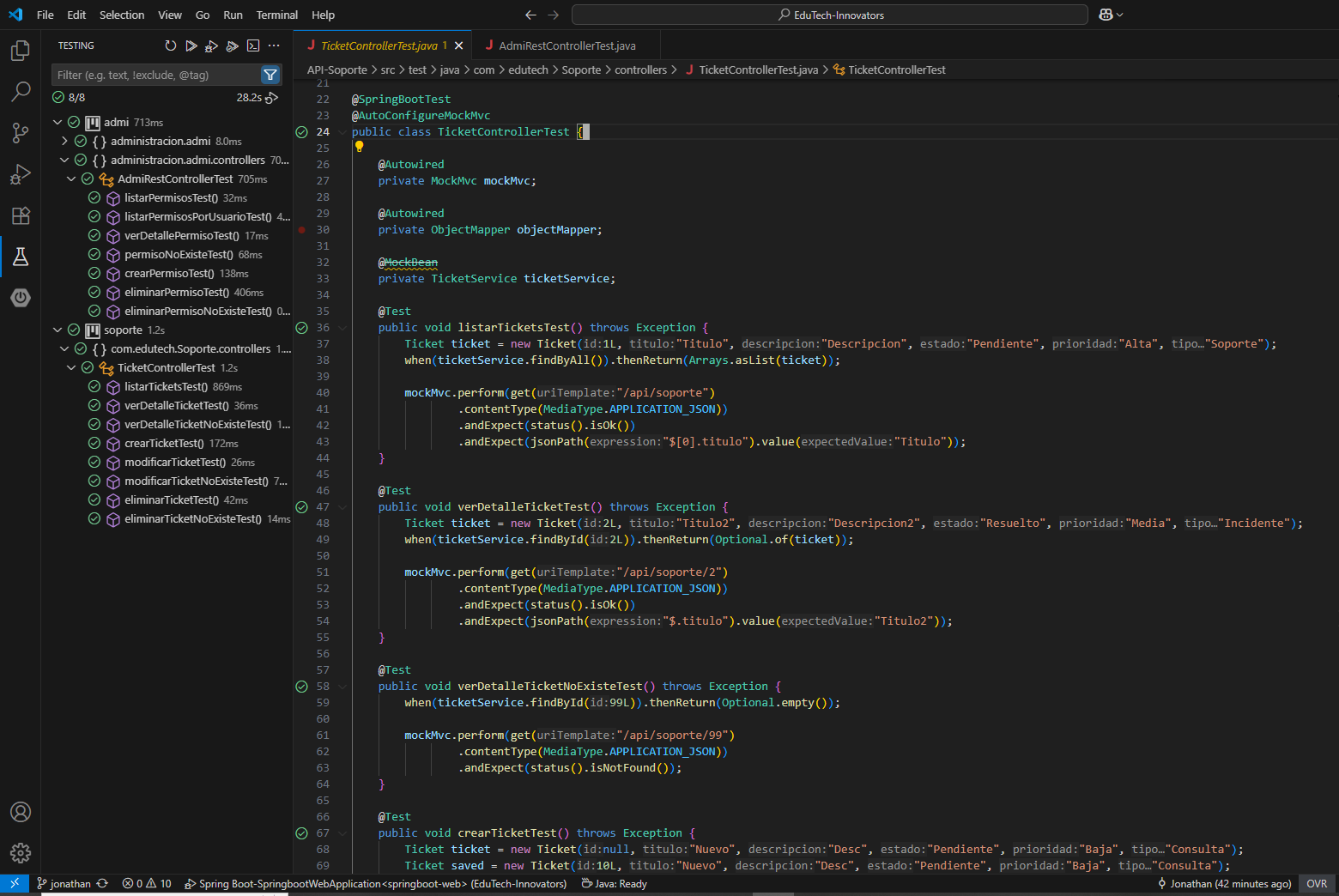
****

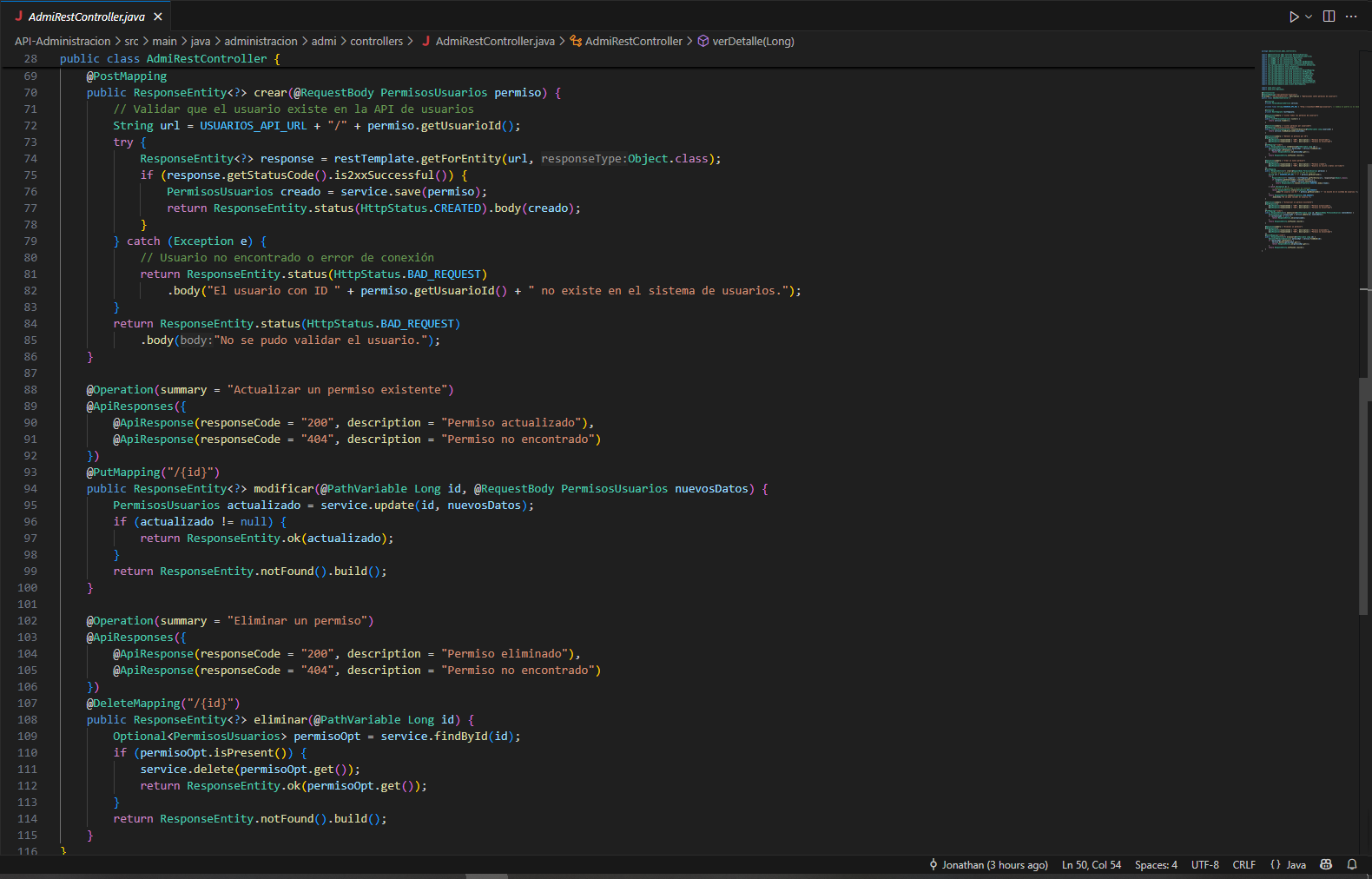
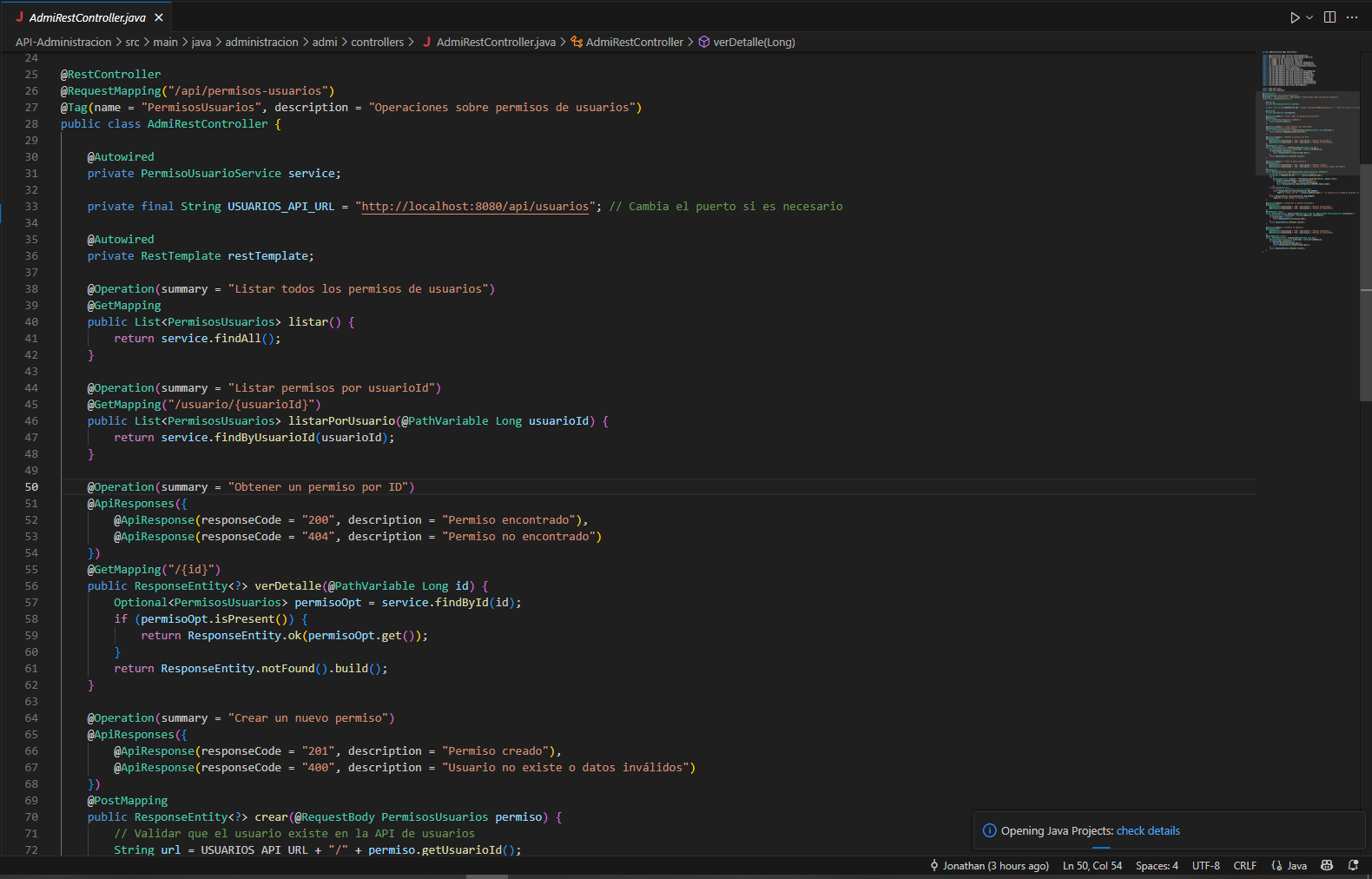
****

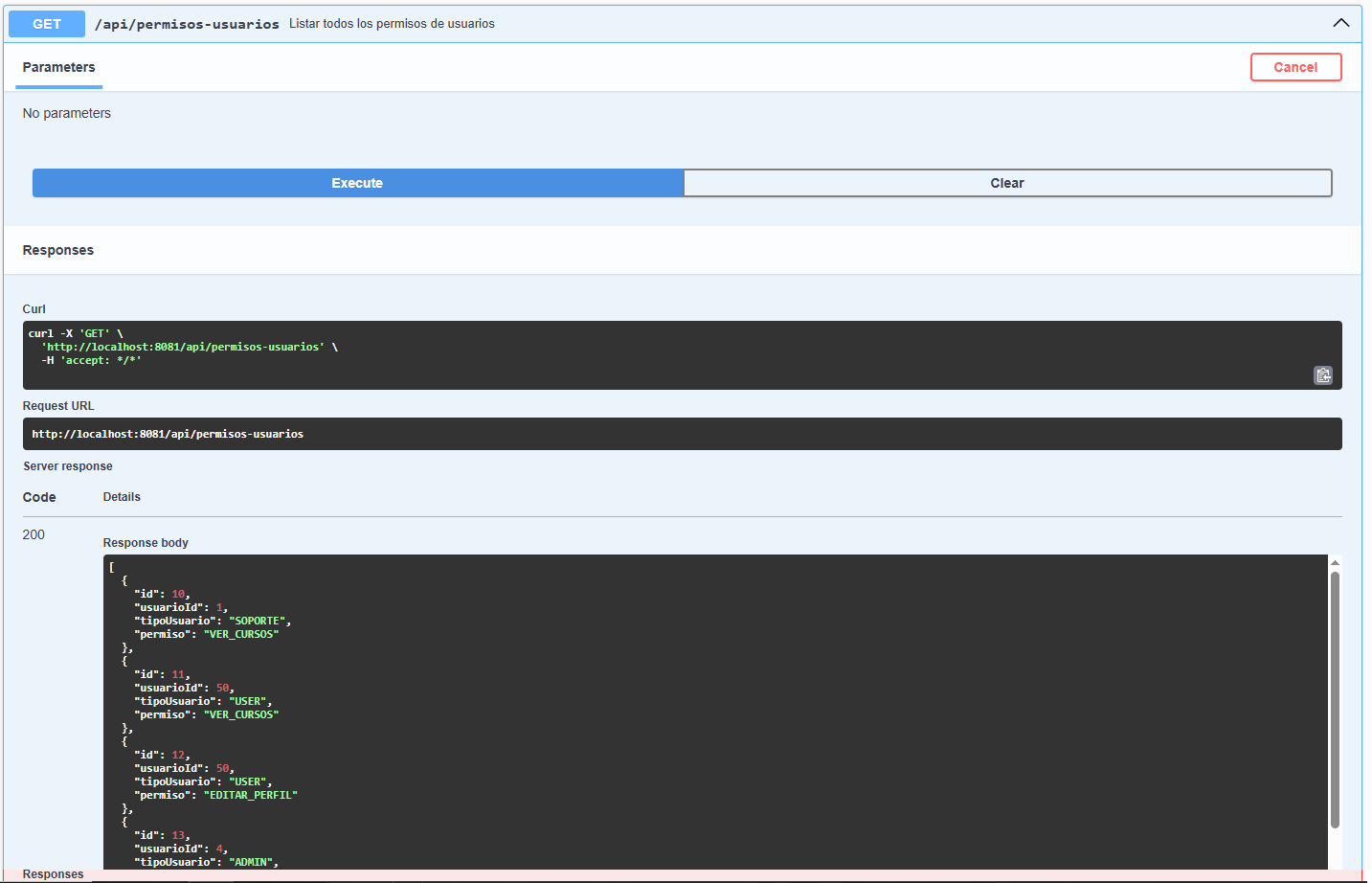
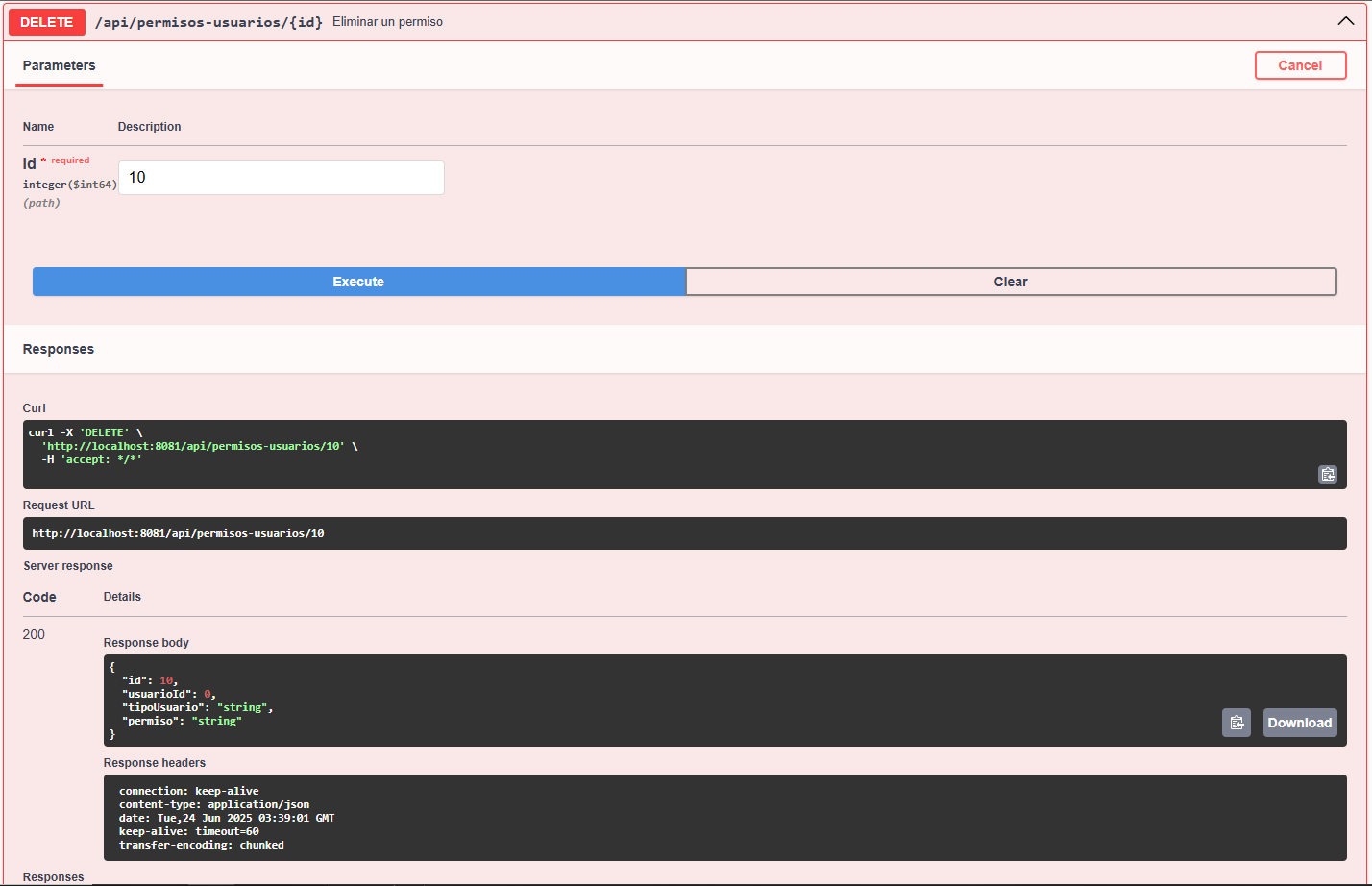
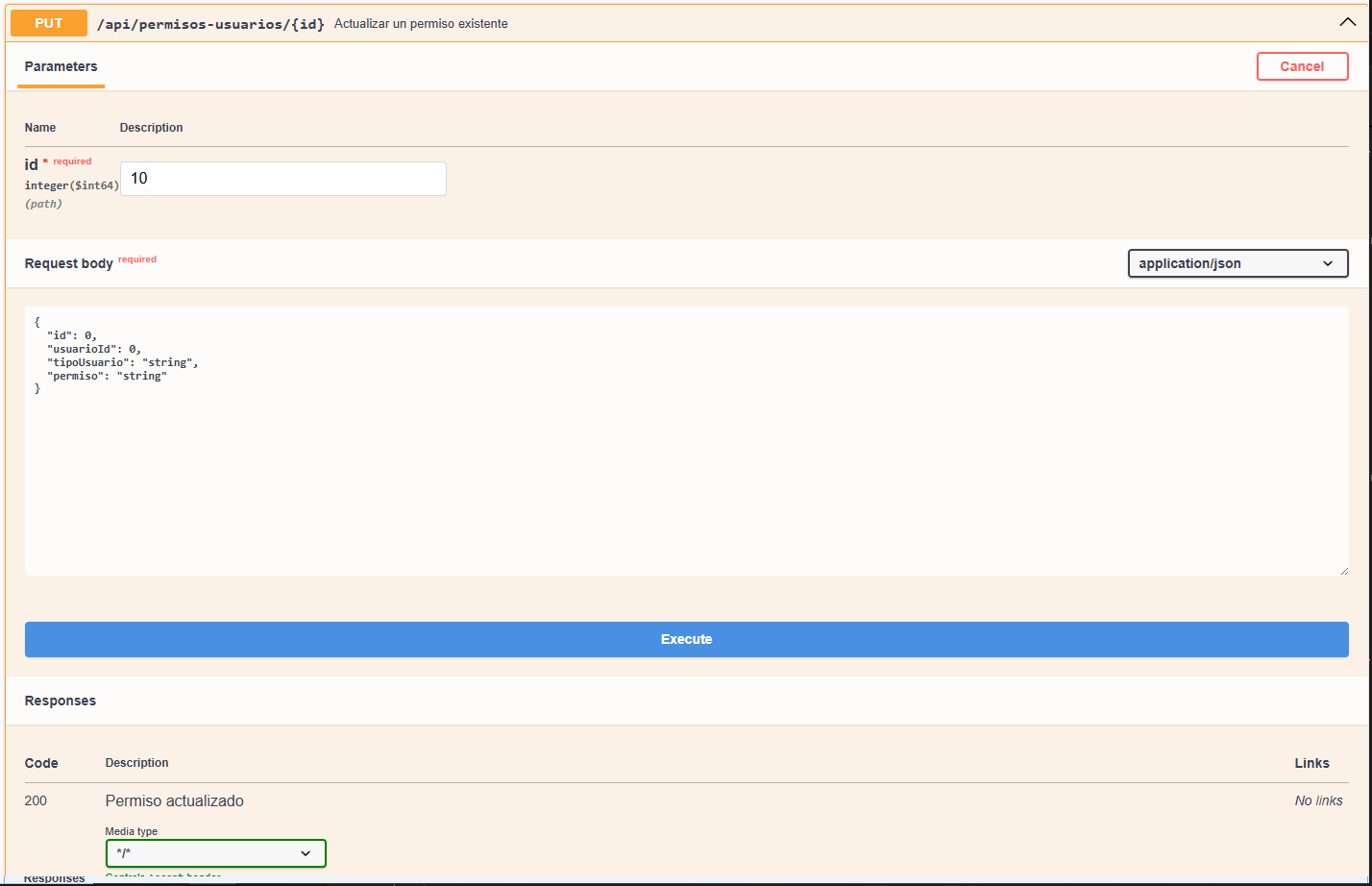
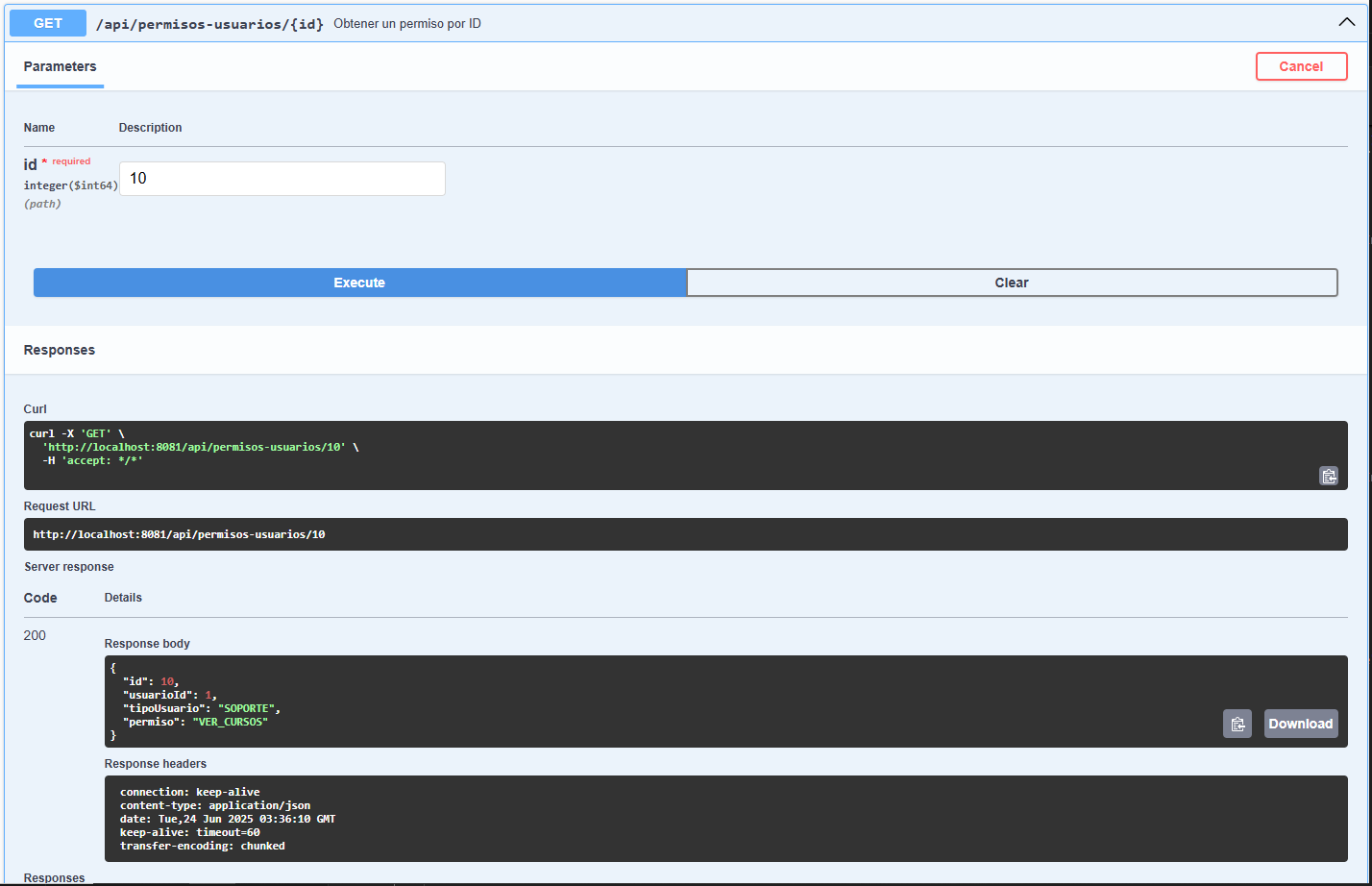
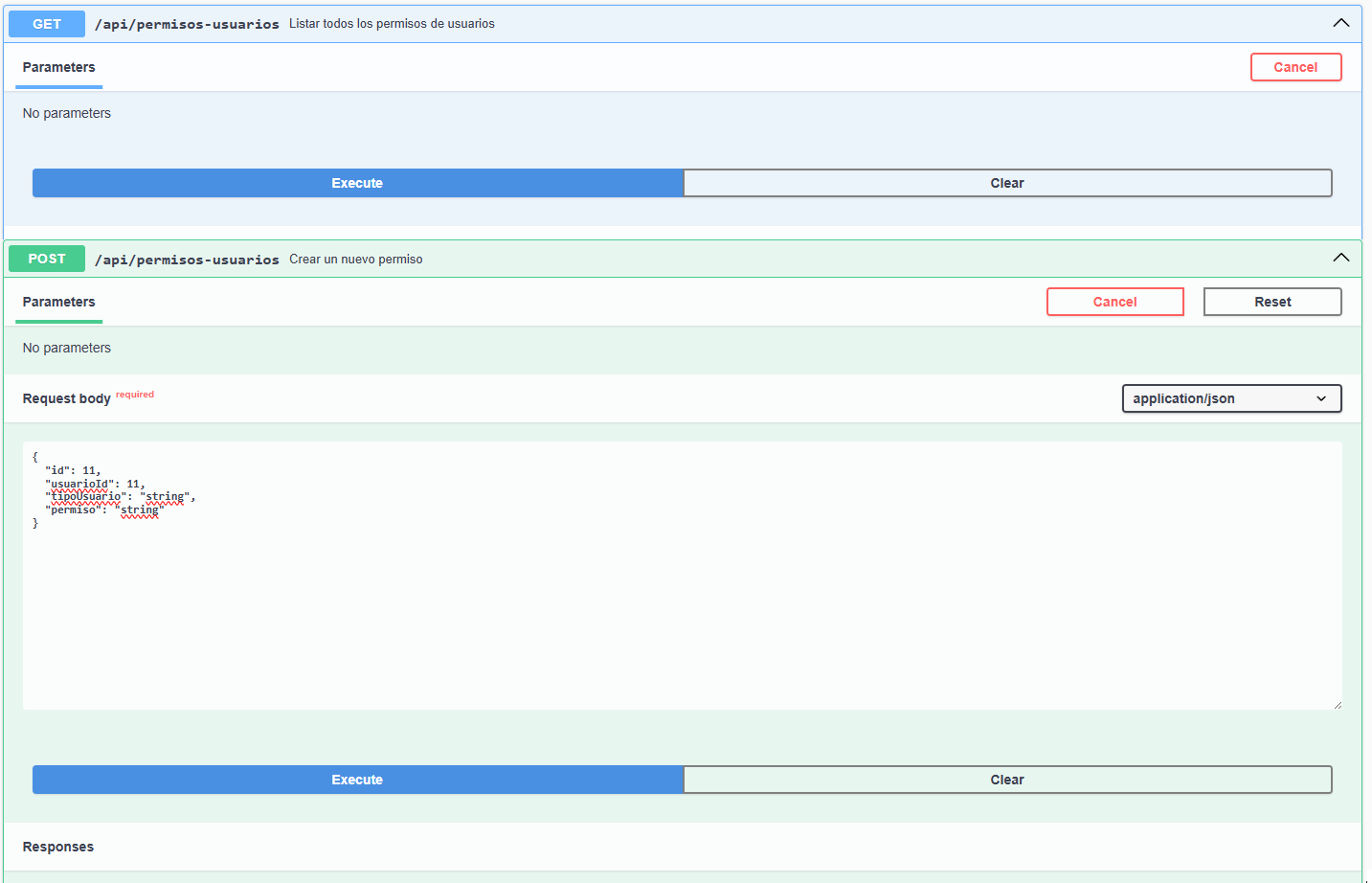
****

****

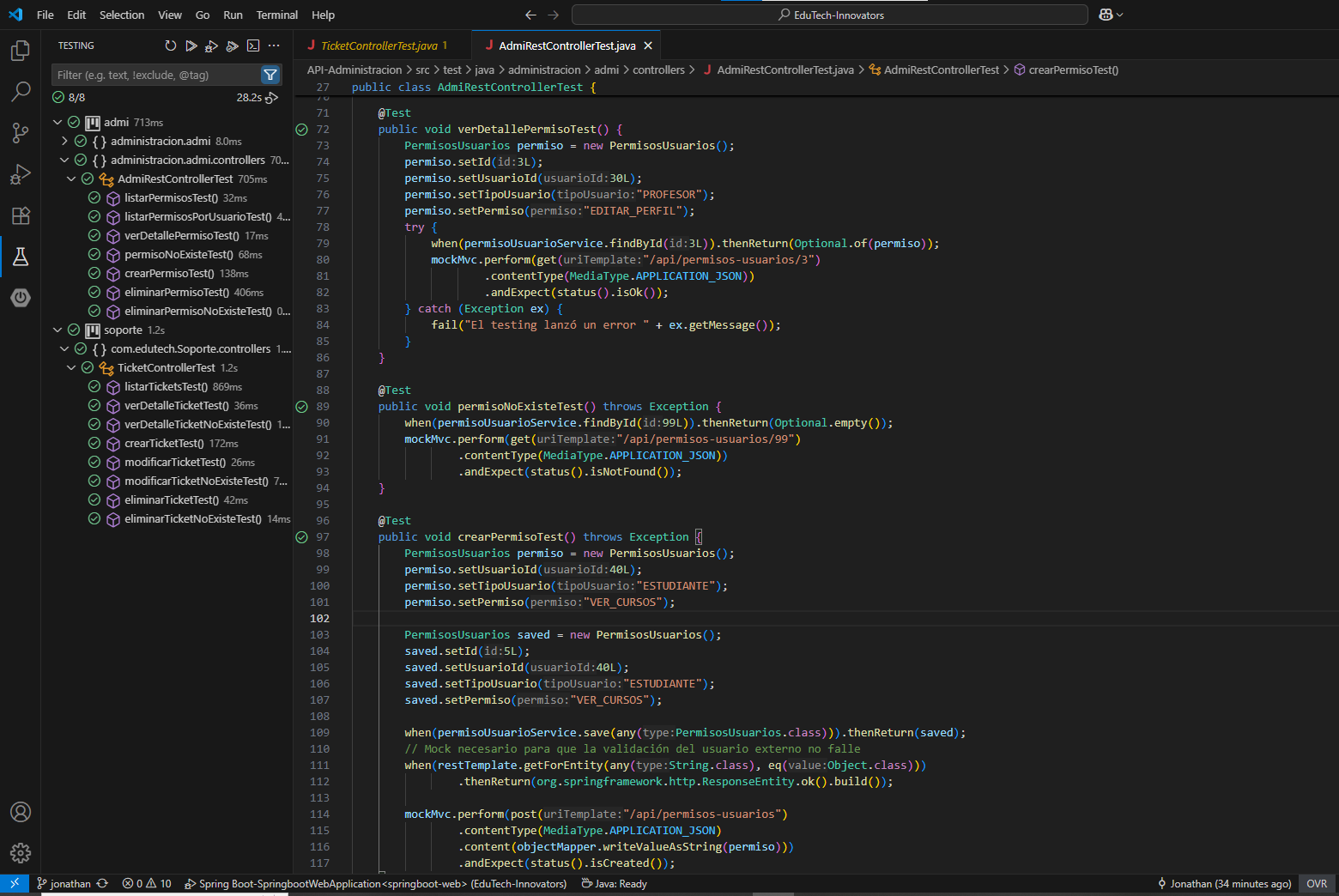
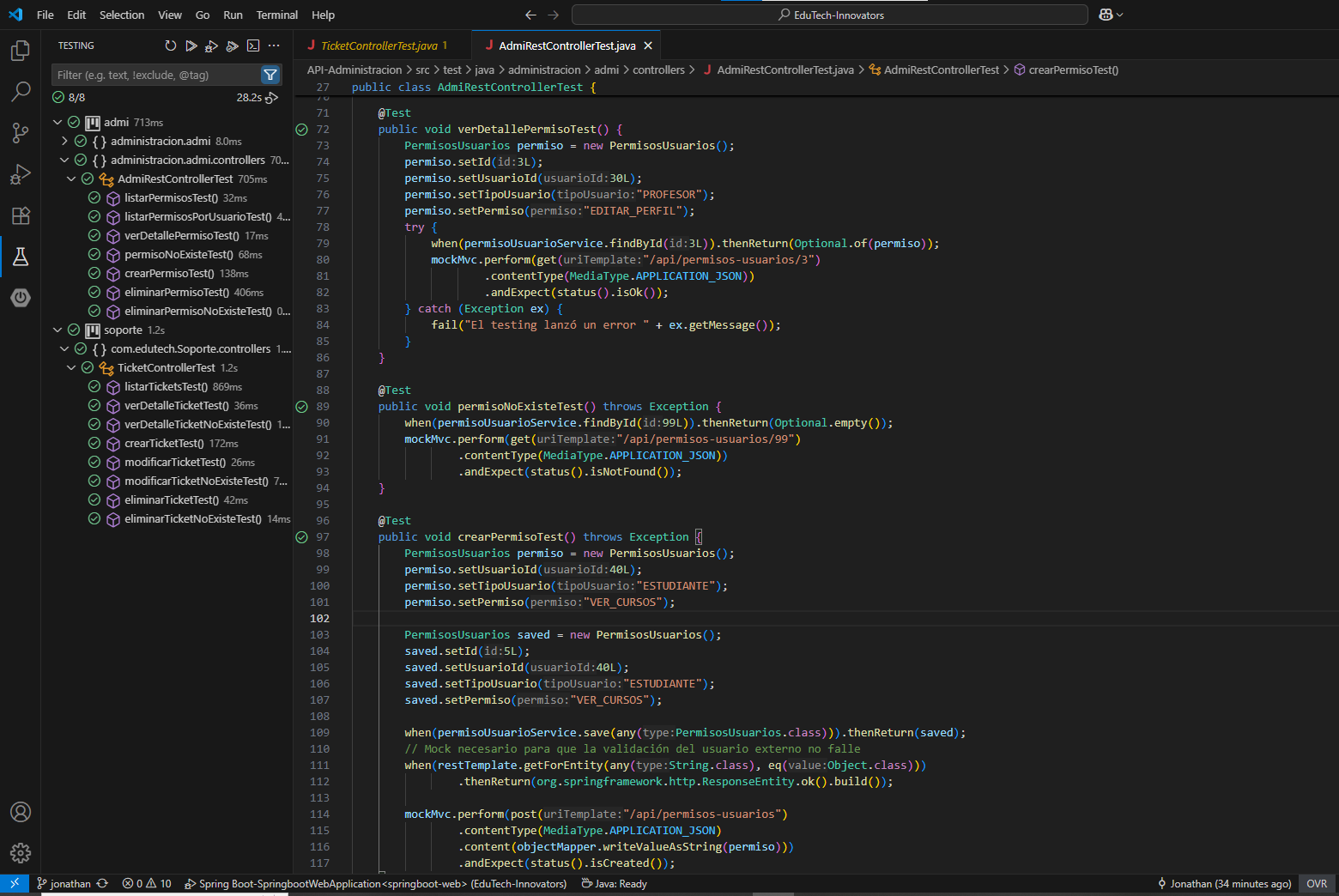
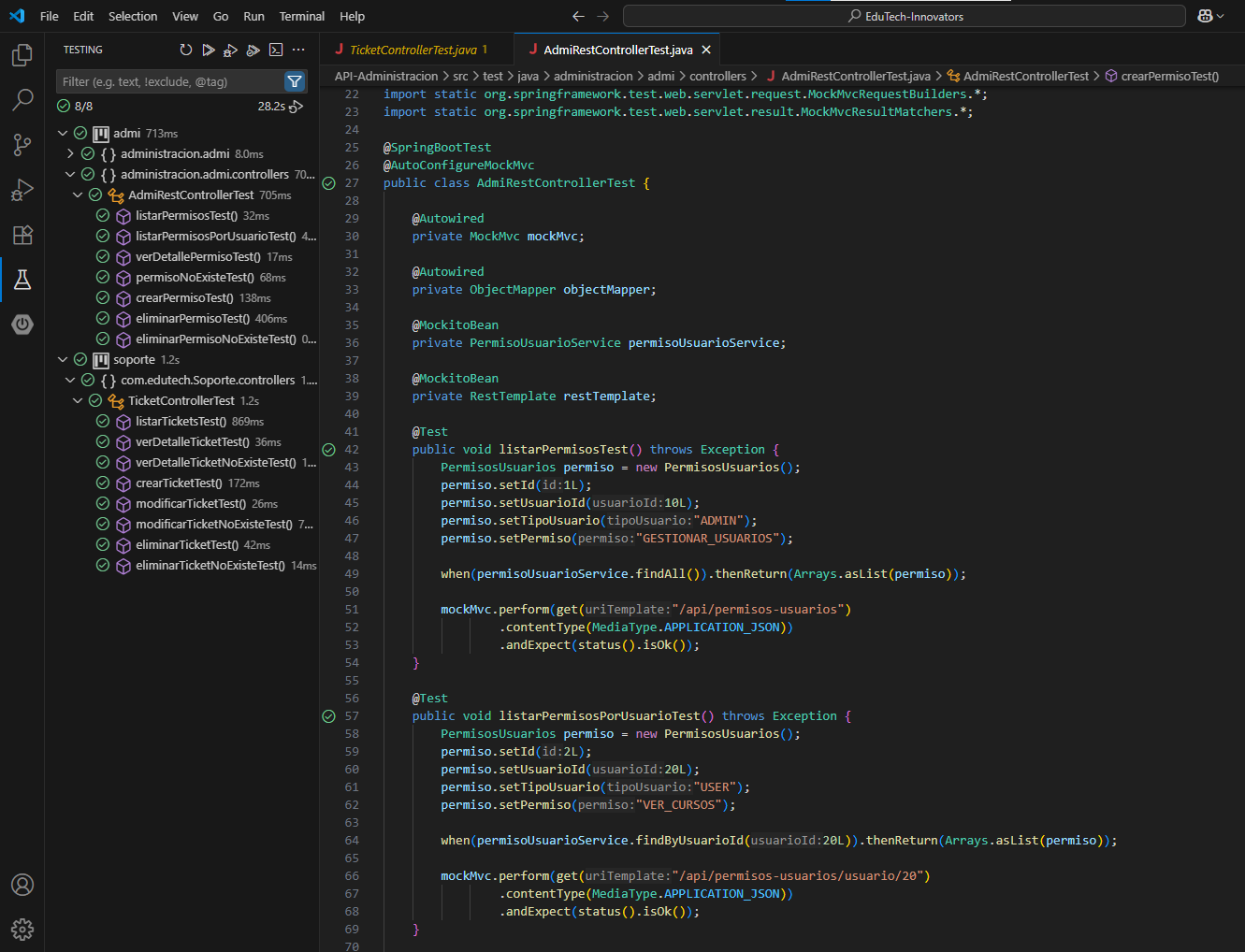
**API-Soporte**

**Pruebas unitarias:**

**Pruebas de integracion OAS:**

****

**API-Administración de Usuarios**

**Pruebas unitarias:**

**Git-GitHub**

## **Proceso de Versionado y Subida del Proyecto a GitHub**

### Clonación del Repositorio

Para iniciar el trabajo colaborativo, se clonó el repositorio remoto desde GitHub mediante el siguiente comando:

git clone https://github.com/dvnxeee/EduTech-Innovators.git

Este comando creó una copia local completa del repositorio en el equipo, incluyendo todo su historial de versiones.

### Exploración de Ramas Remotas

Con el objetivo de identificar las ramas existentes en el repositorio remoto, se utilizó el comando:

git branch -r

Esto permitió visualizar ramas como origin/master y origin/jonathan, las cuales representan las ramas disponibles para trabajar desde GitHub.

### Cambio y Configuración de Rama de Trabajo

Se procedió a cambiar a la rama jonathan con el siguiente comando:

git checkout jonathan

Este comando crea y configura automáticamente la rama local jonathan para rastrear su contraparte remota origin/jonathan.

### Revisión del Estado de Archivos

Previo a confirmar los cambios, se revisó el estado del repositorio local con:

git status

Este comando permite verificar qué archivos fueron modificados, eliminados o añadidos respecto al último commit. En este caso, se identificaron múltiples modificaciones y eliminaciones en carpetas como API-Soporte, API-Usuarios y Proyecto Integrado.

### Confirmación de Cambios (Commit)

Una vez verificados los archivos modificados, se procedió a confirmar los cambios mediante el comando:

git commit -am "Actualización de APIs, eliminación de carpeta antigua y ajustes generales"

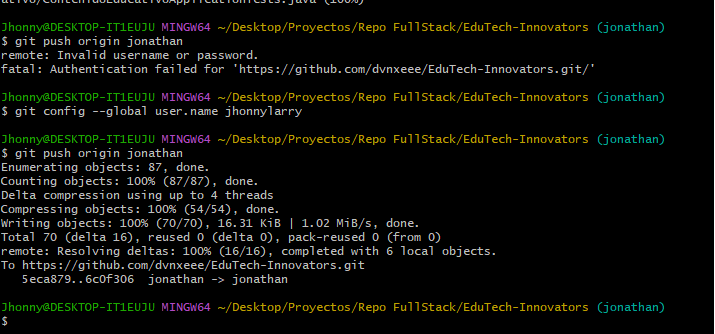
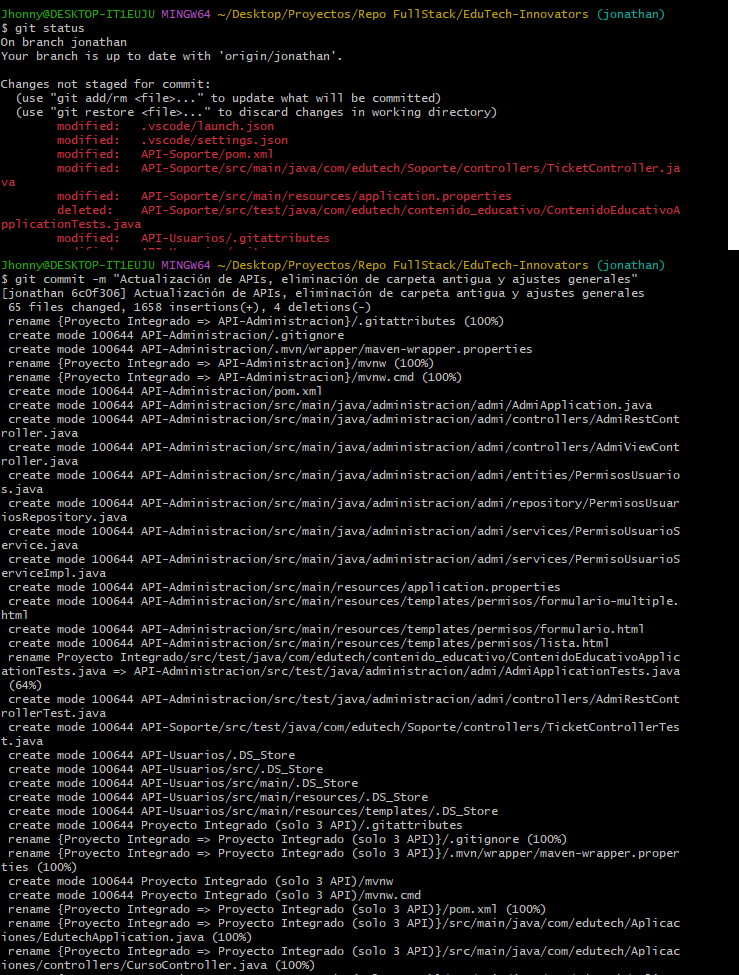
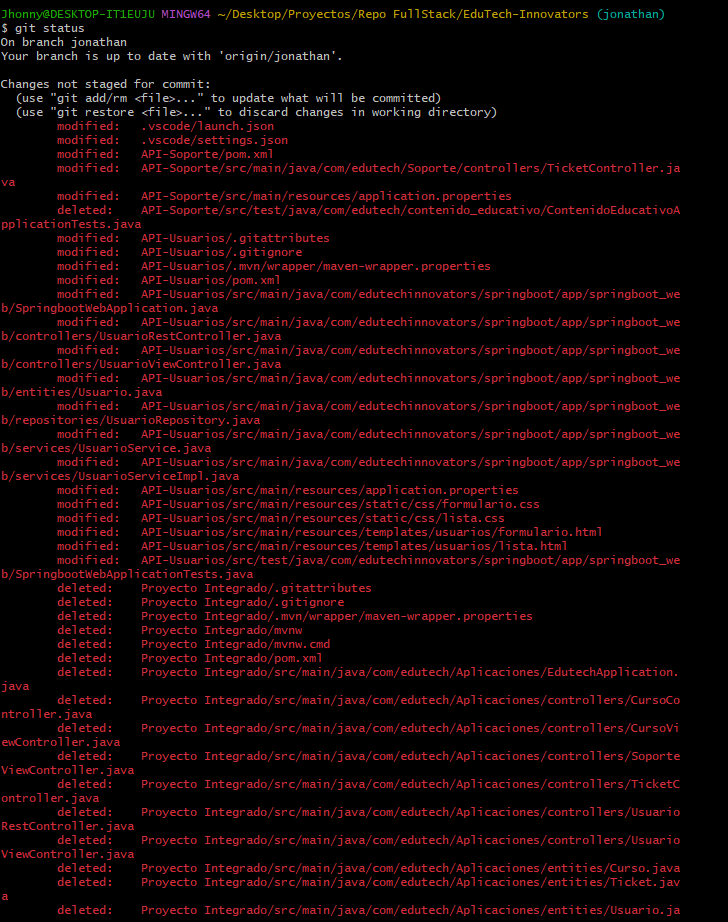
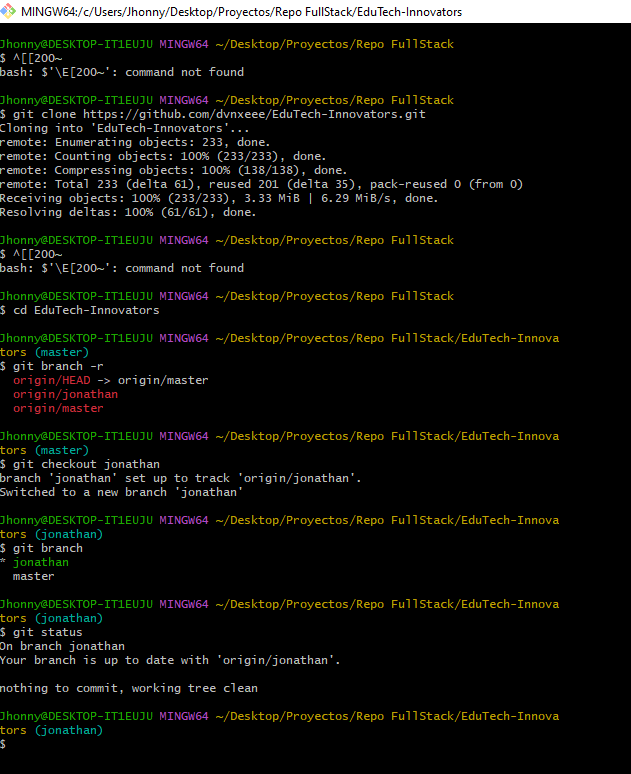
Donde:

* -a incluye todos los archivos modificados y eliminados.
* -m define un mensaje descriptivo que documenta la naturaleza del cambio.

### Subida de Cambios al Repositorio Remoto (Push)

Se intentó subir los cambios a GitHub mediante:

git push origin jonathan En un primer intento, el push falló debido a un error de autenticación. Para solucionarlo, se configuró el nombre de usuario de Git:

git config --global user.name jhonnylarry Luego, el push fue ejecutado nuevamente con éxito, subiendo todos los cambios confirmados a la rama jonathan del repositorio remoto.

# **Conclusión**

El desarrollo del proyecto EduTech Innovators marcó una experiencia integral en la construcción de un sistema educativo modular, estructurado bajo una arquitectura de microservicios y utilizando herramientas modernas como Spring Boot, JUnit, Mockito y GitHub. A través de un enfoque colaborativo y metodologías de pruebas robustas, logramos implementar APIs funcionales, seguras y mantenibles, cumpliendo con los estándares de calidad exigidos en entornos productivos.

La integración de pruebas unitarias e integración, junto con la documentación técnica y el uso adecuado del control de versiones, fortaleció nuestras competencias en ingeniería de software y consolidó una base sólida para futuros desarrollos escalables. Este proyecto no solo permitió aplicar los conocimientos adquiridos en el área de desarrollo full stack, sino también nos preparó para enfrentar desafíos reales en equipos de trabajo distribuidos, asegurando buenas prácticas y una mentalidad orientada a la mejora continua.